University of Nevada, Reno

Intelligent Design for Real Time Networked Multi-Agent Systems

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Electrical Engineering.

by

Sanket Chandan Lokhande

Dr. Hao Xu, Ph.D., Thesis Advisor

December 2017

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

SANKET CHANDAN LOKHANDE

Entitled

Intellingent Design For Real Time Networked Multi-Agent Systems

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Dr. Hao Xu, Ph.D., Advisor

Dr. Yantao Shen, Ph.D., Committee Member

Dr. Hung La, Ph.D., Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

December, 2017

Abstract

This dissertation is the result of two years of work in Autonomous Systems Laboratory at the University of Nevada, Reno. It mainly focuses on Multi-Agent control of drones. Recently, with the advent of powerful embedded processors, drones have become flying resource constrained high-performance embedded control systems. As with any traditional control system, sampling of information is desired at all the times. However, this is not practical. Hence there is need of a strategy where the sampling is done only when required while still maintaining the stability of control system. This thesis presents two such strategies 1) Event-Triggered Consensus Control and 2) Self-Triggered Consensus Control. In addition this thesis presents improvements in these two strategies as presented in first two chapters.

Contributions from this thesis include 1) distributed event-triggered control has been developed for a more general high- order nonlinear Multi-agent systems, 2) The impacts from practical imperfections, e.g. system uncertainties, have been considered, 3) A novel co-model of networked multi-agent consensus control, 4) A novel optimal co-design has been proposed including an admission control scheme from network aspect that allows new agents to be admitted into network based on network usage history and physical system requirement and an optimal control scheme; and 5) Consensus stability is guaranteed using Lyapunov Theory.

I dedicate this thesis to my parents Kalpana and Chandan Lokhande. I hope that this achievement will complete the dream that you had for me all those years ago when you chose to give me the best education possible.

I also dedicate this thesis to my mentor Dr. Daisaku Ikeda and SGI family who have always been there to support me.

Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisor Prof. **Dr. Hao Xu** for his continuous support of my masters study and research, for his patience, motivation, enthusiasm, and immense knowledge.

The work conducted in this dissertation has been developed in Autonomous Systems Laboratory of University of Nevada, Reno. I would like to thank **Dr. Eduardo Steed** from **Polytechnic University of Pachuca, Mexico** and **Dr. Luis Rodolfo Garcia Carrillo**, from **Texas A&M University - Corpus Christi** without whom the Autonomous Systems Laboratory would have been far afield from reality. Their work and vision for Multi-Agent drone systems is prescient.

I would also like to thank **Zach** and **Abdelghafor** for providing me with the transportation for outdoors experiments.

Finally I want to thank our department technician **Anthony Piazza**, at UNR for his resoursefullness and expertise.

Sanket Chandan Lokhande

University of Nevada, Reno

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Past decade has witnessed an unprecedented growth in reasearch for Unmanned Aerial Vehicles (UAVs) both in military and nonmilitary fronts. They have become ubiquitous in almost every military operations which includes domestic and overseas missions. With rapidly advancing technology, open source nature of the flight controllers, and significantly lesser costs than before, companies around the world are delving into UAV market as one of the upcoming lucrative investments. Companies like Amazon Inc., Dominos Pizza Inc. have had some successful test runs which again solidifies the research opportunities. Delivery services and recreational uses seems to have increased in the past 3-4 years which has let the Federal Aviation Administration to update their rules and regulations. Mapping, Surveying and search/rescue mission are some of the applications of UAVs that are most appealing. Making these applications airborne cuts the time and cost at considerable and affordable levels.

Using UAVs for operations has advantages in both response time and need of manpower compared to piloted aricrafts. Obtaining prior information of a person/people in distress can become a deciding factor for a successful mission. It can

help in making critical decision as which location or type of helicopter / vehicle to be used for extraction, equipment to bring and how many crew members that are needed. The idea here is to make this system of UAVs automated to coordinate with each other without human intervention (other than high level commands like takeoff and land).

Researchers and Military experts have recognized the use of drones for search and rescue missions to be of utmost importance. Year 2016 saw a first of its kind UAV search and rescue symposium held in Nevada. The objective was to give a platform for UAV enthusiasts and researchers and share their experiences and concerns while using UAVs as first responders.

The biggest drawback of using an aerial vehicle for inspection/search/rescue mission is its airborne time. The batteries used are big and heavy which increases the weight and decreases the flight time. One can go about solving this issue by using a swarm of UAVs which would inspect/search a given area in less amount of time. This has advantage in both response time and need for lesser man power.

1.1 Problem Description

The main challenges for Multiple Drone Control (MDC) includes 1) Address the periodic sampling frequency issue of information of assets so as to maintain stability; 2) Optimize the communication channel while providing minimum Quality of Service (QoS); 3) Optimal control strategy which includes non-linearity in state space model; 4) Optimal control in presence of uncertainties; 5) Admitting new agents for dynamic agents in the Networked Multi-Agent System (MAS) Scenario.

1.2 Contributions

This dissertation aims at building a hardware and a software platform for communication of multiple UAVs upon which additional control algorithms can be implementated. It starts with building a DJI S1000 octacopter from the ground up. The components used are specified in the following sections. The idea here is to make a drone that can autonomously travel to specified location with safety features like geofencing and land on emergency situations. The user has to provide the necessary commands like GPS locations and takeoff/land commands via a Radio Controller (RC) remote. At any point of the flight, the UAV should be able to receive new commands from the ground control stations (GCS). After successful implementation, the UAV would not be restricted to the range of RC remote. It would be able to travel greater distances given the GPS signal remains operational in the field. This is possible at a global scale with limitation of only the batteries and flight time.

Following figure 1.1 shows the scope and contributions according to the chapters of this thesis.

1.3 Related Work on communication of MAS

There are many videos and online resources which are helpful in making a drone from the ground up. The uploading of firmware on to pixhawk controller is made fairly straight forward and minimalistic knowledge of programming on the users part.

Streamlining the related work and the purpose of this dissertation, Multiagent control, quiet a few researchers have been successful in implementing a swarm of agents to collaborate and complete specific missions. In [1] Vijay Kumar et.al

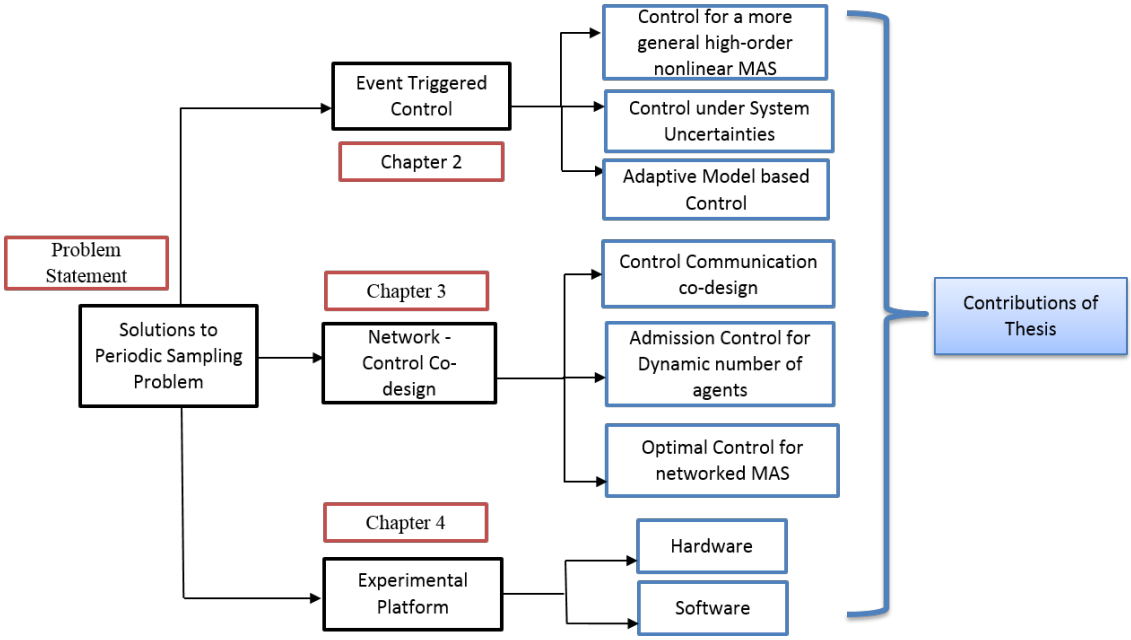

Figure 1.1: Scope of Thesis

have implemented a communication system for multiple robots. Similar strategy is presented here and UAVs are taken into account.

In addition, Miguel Duarte et.al. in Evolution of collective behaviors for a real swarm of aquatic surface robots, presents for the first time, a swarm of surface water bots collaborating for consensus in heading and positions. They present a mathematical model, their simulations and a proof-of-concept experiment which is performed in real time. This thesis presents a similar approach in delving into the problem statement and identifying the non-linearities involved in real time experiments.

Chapter 2

Event Triggered Distributed Adaptive Consensus Control for High-order Nonlinear Multi-Agent Systems in presence of system uncertainties

2.1 Abstract

In this paper, a novel event-triggered distributed adaptive consensus control strategy has been developed for high-order nonlinear multi-agent systems (MAS) with uncertain system dynamics. Compared with most existing results, proposed approach contributes on two main aspects, i.e. 1) distributed event-triggered control has been developed for a more general high-order nonlinear MAS, and 2) The impacts from practical imperfections, e.g. system uncertainties, have been considered. Firstly, the system uncertainties are considered as bounded with a given upper bound. Through incorporating the upper bound

of system uncertainties, a novel distributed event-triggered condition has been developed that can maintain the consensus of uncertain nonlinear MAS. Then, the system uncertainties are considered as unknown completely. A novel event-triggered distributed adaptive consensus control has been developed. For handling the unknown system dynamics, a series of novel distributed adaptive estimators have been designed to learn the system dynamics by using measured system data. Then, using the learnt system, the distributed event-triggered condition can be attained. Lyapunov stability analysis have been adopted to demonstrate the stability of developed novel event-triggered distributed consensus control. Eventually, simulation results are provided to verify the effectiveness of proposed designs.

Keywords- Event Triggered, Consensus Control, System uncertainities, High order Nonlinear, Multi-agent, Distributed Adaptive.

2.2 Introduction

For the past few decades, research on time-based periodic sampling scheme has been dominant in most aspects of modern feedback control such as sample-data system [1], multi-agent systems (MAS) [2-3] and so on. However, with the appearance of more and more distributed control systems, there is a demand for time-based periodic control scheme which requires suitable communication bandwidth to transmit the feedback information between sensors, controllers and actuators [4]. This in turn lead to insufficient network resources in the practical MAS [5-6]. Therefore, event-driven sampling and feedback control methodologies [7-8] have been developed as an alternative to the time-driven techniques for reducing the communication burden. The main idea behind these schemes is that the sampling of

system states and executing control inputs are based on an event triggering condition which is derived by using MAS system states [8]. The input-to-state stability (ISS) property of event -triggered control scheme is analyzed by [9-10] by deriving an event-trigger condition which is based on the norm of the error of the current and the most previous received state vector at the controller. In [9], the controller kept the latest received system state vector from the sensor until a new system state vector is received when the event trigger condition is not satisfied. After receiving a new state vector, the measured state error is reset to zero until the next event trigger condition is not satisfied. In [13], the authors introduced a new trigger condition to achieve optimality by using optimal stopping theory [14]. Until now, the schemes in [9] [13-14] are usually classified as Zero-Order-Hold (ZOH) event-triggered control schemes since measured state and designed control inputs are held constant during two triggered events. The system matrices are needed to design a stabilizing gain matrix. The event-triggered control schemes and stability analysis have been considered for centralized and decentralized control system [4], [11-12]. Garcia and Antsaklis [19] introduced a model-based event-triggered control scheme for continuous-time system by using a known system model. Here, the model, in the form of a state estimator or observer, generates the estimated system states and associated trigger condition which is shown to save more communication resources over ZOH-based event triggered control schemes. Despite these similar works [9-14] [19], the event triggering condition and associated controller designs are addressed for continuous-time linear systems needing explicit knowledge of system dynamics [9-14] or using a model with bounded uncertainties [19]. Nevertheless, the studies mentioned earlier are limited in linear multi-agent environment with bounded system models. However, this is not true for many real world applications as they exhibit non-linear behavior. Hence study of non-linear

behavior and non-linear system models becomes important in multi-agent systems [29]-[30]. Aforementioned non-linear studies deal with bounded systems and do not delve into system uncertainties. This is especially true when heterogeneous multi-agent system is deployed on field.

Therefore, this paper presents a novel adaptive model-based event-triggered consensus control scheme and associated stability and consensus analysis for heterogeneous high order non-linear MAS with system uncertainties.

Thus the main contributions of this paper include: 1) a novel distributed event-triggering condition and associated distributed consensus control design for MAS; 2) an online non periodically tuned distributed adaptive model-based event-triggered consensus control scheme to relax the requirement of full system dynamics and to construct uncertainties online; and, 3) the demonstration of the closed-loop stability in the presence of system uncertainties. Compared with [19], our proposed scheme can relax the assumption that model uncertainties need to be small [19] and reduces more network traffic than [8] [19] after the model uncertainties are estimated. It is found from this work that when the MAS dynamics are uncertain, the MAS states have to be transmitted more often until the information about the system dynamics is constructed online.

2.3 Background

2.3.1 Graph Theory

We will state some formal definitions about algebraic graph theory. One can find detailed explanation in [16] The connectivity of MAS is generally represented through using graph theory [11]. Specifically, an undirected graph is expressed as $G = \{V, E\}$, where $V = \{1, 2, ...N\}$ represents a set of vertices with N being the last

vertex, and the related edge set E defined as $E \subseteq \{(i,j) : i,j \in V, i \neq j\}$. Then, the edges can be used to represent a bidirectional communication links in MAS where vertices are distributed agents. Similar to [4], the adjacency matrix $A(G) = (a_{ij})$ $\forall i,j = 1,2,...N$, with element a_{ij} defined as $a_{ij} = 1$ if and only if $(i,j) \in E$ and $a_{ij} = 0$, otherwise. The communication graph can be defined as connected when there is a path connecting each pair of distinct vertices. Each agent in the MAS is assumed to have equal omni-directional communication and sensing capability which indicates that there is a mutual communication within the connected MAS. Mathematically, the adjacency matrix is symmetric, i.e. $A^T = A$, and the communication graph is undirected. An undirected communication graph topology example is demonstrated in Figure 1.

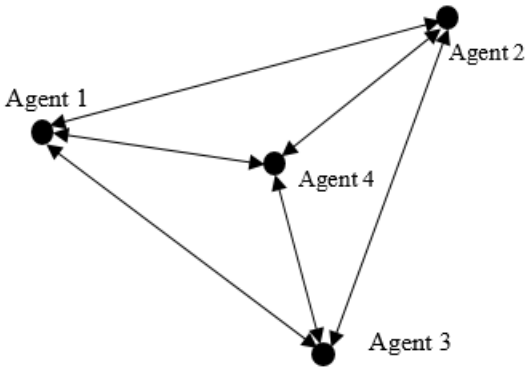

Figure 2.1: An undirected Graph representing communication between 4 agents

Moreover, the degree matrix of communication graph G is defined as $D = diag\{A\}$ where the diagonal element d_{ij} is given as $d_{ij} = \sum_{j=1,j\neq i}^{N} a_{ij}$. Further, The Laplacian matrix $L = \{l_{ij}\} \in \Re^{N\times N}, \forall i,j = 1,2,....N$, can be represented as $L = D - A$.

2.3.2 Distributed Event-triggered Consensus Control

Consider a system of multiple agents governed by the control law as follows:

$$\dot{x}_i(t) = f_i(x) + u_i(t) + d_i(t) \qquad \forall i, j = 1, 2, ...N \tag{2.1}$$

where $x_i(t), f_i(x), g_i(x) \in \Re^N$, $u_i(t) \in \Re^M$, $d_i(t)$are the system states, a non-linear function, an arbitrary function based on system states and control inputs for agent i, and disturbance respectively.

Based on [14], the control input can be designed as follows

$$u_i(t) = -\hat{f}_i(x) - \sum_{ij=1, i\neq j}^{N} a_{ij} \left(x_i(t) - x_j(t)\right) \qquad \forall i, j = 1, 2, ...N \tag{2.2}$$

where a_{ij} denotes the connectivity between agent i and agent j [14] and $\hat{f}_i(x)$ is uncertainty. Note: the uncertainty and disturbance are bounded, i.e. $\left\| f_i(x) - \hat{f}_i(x) \right\| \leq \Delta_i$, and$\|d_i(t)\| \leq d_m$ Next we consider the event-trigger control scenario. Here we distinguish the input in two different cases. One where the event has been triggered and the other where event is not yet triggered. For both the cases, state $\hat{x}_i(t)$ can be defined as follows:

$$\hat{x}_i(t) = \begin{cases} x_i(t), & if\ event\ is\ triggered \\ x_i(t_k), & if\ event\ is\ not\ triggered \end{cases} \qquad \forall i, j = 1, 2, ...N \tag{2.3}$$

where $x_i(t_k)$ is the latest triggered event for agent i at time t_k. Further the error measurements can be expressed as

$$e_i(t) = \hat{x}_i(t) - x_i(t) \qquad \forall i, j = 1, 2, ...N \tag{2.4}$$

From (4), it can be inferred that when the event is triggered the error will go to zero as the system is expected to transmit the information between agents as specified by the adjacency matrix. Using (1), (2), (3) and (4) the event triggered input control $u_{ET,i}(t)$ can be expressed as follows:

$$u_{ET,i}(t) = -\hat{f}_i(\hat{x}_i(t)) + \hat{x}_i(t) - \sum_{j=1}^{N} a_{ij}(x_i(t) - x_j(t)) \qquad \forall i, j = 1, 2, ...N \tag{2.5}$$

The measured state and control inputs are held to their values between any two triggered events, state measurement errors might increase quickly in Zero Order Hold event-triggered control scheme. To overcome this shortcoming a model based scheme is introduced in [15] where a model-based estimator is employed to extrapolate missing data points between events. However, [15] proposed a fixed model based estimator which is not a practical approach. Therefore, in this paper we present an adaptive model based estimator. In figure 2, the block diagram describing the adaptive model-based event-triggered consensus control is shown. Here the adaptive estimator is responsible for extrapolating the missing control inputs between events.

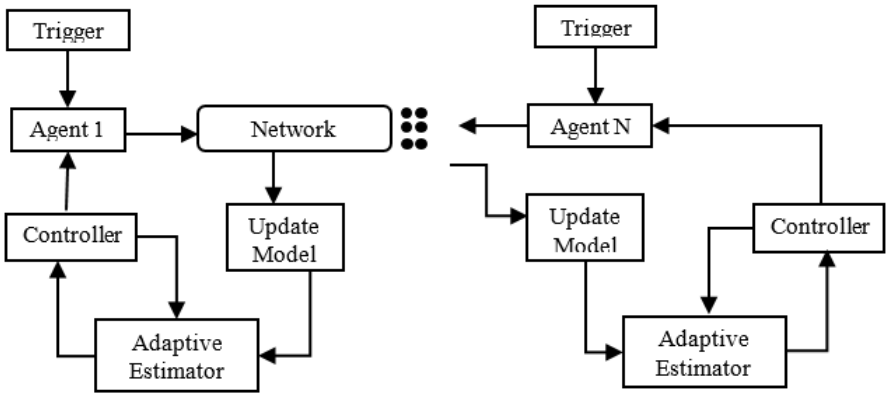

Figure 2.2: Adaptive model-based event-triggered control system

2.4 ZOH and fixed model-based event-triggered control design

In this section traditional ZOH event triggered control for a system with bounded uncertainty is discussed. Moreover, it is assumed that the communication delay is negligible and the initial conditions of the system are non-zero and finite.

2.4.1 ZOH Distributed Event-Triggered Consensus Control

Different from periodic sampling scenario, the ZOH distributed event-triggered consensus controller might not receive the system state at every sampling time instant. Hence the distributed consensus controller will hold the latest received local system state and its neighbors latest received vector for control input design until the new local state vector and its neighbors new state vector are received due to an event caused by condition when local state measurement error $e_{i,k}$ exceeds the threshold. Now, consider the heterogeneous MAS (1), and the ZOH distributed event-triggered consensus control for agent i is given by

$$u_i(t) = \begin{cases} -\hat{f}\left(x_i(t)\right) + x_i(t) - \sum_{j \in N_i} a_{ij}\left(x_i(t) - x_j(t)\right) \\ \qquad when\ event\ not\ triggered \\ -\hat{f}\left(x_{i,ET}(t)\right) + x_{i,ET}(t) - \sum_{j \in N_i} a_{ij}\left(x_{i,ET}(t) - x_{j,ET}(t)\right) \\ \qquad when\ event\ triggered \end{cases} \qquad \forall i, j = 1, 2, ...N \tag{2.6}$$

Here and are states of agent i and agent j respectively. Moreover, the states are the latest local state measurements at time t. These states remain constant until a new measurement update cycle is triggered. The new update can be denoted by . The

consensus error is represented by following equation:

$$z_i{}^{ZOH} = \begin{cases} 0, & if\ event\ is\ triggered \\ L \times \sum_{i,j \in N_i} (x_i(t) - x_j(t)),\ if\ event\ is\ not\ triggered \end{cases} \quad \forall i, j = 1, 2, ...N \tag{2.7}$$

After substituting the control input $u_i(t)$ from (6) and (1), the closed-loop system dynamics due to the ZOH distributed event-triggered consensus control input can be expressed as

$$\dot{x}_i(t) = f_i\left(x_i(t)\right) - \hat{f}_i\left(\hat{x}_i(t)\right) - \sum_{j=1}^{N} a_{ij}\left(\hat{x}_i(t) - \hat{x}_j(t\ )\right) + d_i(t) \quad \forall i, j = 1, 2, ...N \tag{2.8}$$

Obviously if the hold time of the ZOH distributed event-triggered consensus control system is too large the MAS will not be stable. To deduce the stability of the MAS we prove that the error has to be less than a certain threshold value. Following theorem shows error threshold condition and the proof is also stated. In this paper $\|\bullet\|$ denotes Frobenius norm.

Theorem 1 *(ZOH Distributed Event Triggering Condition for multi agent system consensus)*

In distributed event-triggered multi agent system (1), the event should be triggered and controller should be updated when the following is satisfied

$$\|e_i(t)\|^2 \leq \frac{\sigma_i\left(1 - l_1 N_i - l_2\right)\|z_i(t)\|^2}{\left(\frac{1}{l_1}N_i + \frac{l_i^2}{2l_2}\right)} \quad \forall i, j = 1, 2, ...N \tag{2.9}$$

where $z_i(t) = L \times \Sigma\left(x_i(t) - x_j(t)\right)$ with $0 < \sigma_i < 1$ and $l_1 < \frac{0.9}{N_i}$,$0 < l_2 < 1 - l_1 N_i$

Proof: Consider the quadratic Lyapunov function candidate as $V_{\ ZOH}(t) = x^T(t)Lx(t)$. Using system representation 1 and Cauchy-Schwartz inequality, the first difference of Lyapunov function candidate can be derived as fol-

lows

$$\begin{aligned}\dot{V}_{ZOH}(t) &= x^T(t)Lx(t)\\ &= x^T(t)L\left[-Lx(t) - Le(k) + f(e(k)) + \tilde{f}(\hat{x}(t)) + d(t)\right]\\ &= -z^T(t)z(t) - z^TLe(k) + z^Tf(e(k)) + z^T\left[\tilde{f}(\hat{x}(t)) + d(t)\right]\\ &= -\sum_{i=1}^{N}\|z_i(t)\|^2 - \sum_{i=1}^{N}N_iz_i^T(t)e_i(t) + \sum_{i=1}^{N}\sum_{j\in N_i}z_i^T(t)e_j(t)\\ &\quad + \sum_{i=1}^{N}z_i^T(t)f_i(e_i(t)) + \sum_{i=1}^{N}z_i^T(t)\left[\tilde{f}_i(\hat{x}_i(t)) + d_i(t)\right]\end{aligned} \tag{2.10}$$

Through applying the event trigger condition (6) with $0 < \sigma_i \leq 1$ and then ΔV_{ZOH} is negative definite while V_{ZOH} is positive definite. Therefore, the ZOH distributed event-triggered closed-loop multi agent system is globally asymptotically stable and consensus. In other words, as $t \to \infty$, $x_i \to 0, x_i \to x_j, \forall i, j = 1, 2, ..., N$, $i \neq j$.

Now to satisfy the stability condition according to Lyapunov Theory, $V_{ZOH}(t)$ should be a negative semidefinite, hence

$$\begin{aligned}\Delta V_{ZOH} \leq &-\sum_{i=1}^{N}\|z_i(t)\|^2 + \sum_{i=1}^{N}l_1N_i\|z_i(t)\|^2 + \sum_{i=1}^{N}\frac{1}{2l_1}N_i\|e_i(t)\|^2\\ &+\sum_{i=1}^{N}\sum_{j\in N_i}\frac{1}{2l_1}\|e_j(t)\|^2 + \sum_{i=1}^{N}l_2\|z_i(t)\|^2\\ &+\sum_{i=1}^{N}\frac{1}{2l_2}\|f_i(e_i(t))\|^2 + \sum_{i=1}^{N}\frac{1}{2l_2}\left\|\tilde{f}_i(\hat{x}_i(t)) + d_i(t)\right\|^2\end{aligned} \tag{2.11}$$

Finally,

$$\Delta V_{ZOH} \leq -\sum_{i=1}^{N}(1 - l_1N_i - l_2)\|z_i(t)\|^2 + \sum_{i=1}^{N}(\frac{1}{l_1}N_i + \frac{l_i}{2l_2})\|e_i(t)\|^2 \tag{2.12}$$

Remark 1: In ZOH, the system is running closed-loop based on control input de-

rived by using the local system state and its neighbors system state that are received previously.

2.5 Adaptive model based event-triggered consensus control for uncertain system

In the adaptive model-based event-triggered consensus control scheme, the model parameters are constantly adjusted once per event when the MAS consensus controller receives the corresponding MAS system state vector from the sensor. At the update times, the MAS state vector of the MAS model is also updated with the measurements obtained from local system; the update times are non-periodic in general and are triggered by the size of the MAS state estimation error. As a consequence, the model update is also not periodic in contrast with traditional adaptive control literature [15]. With an adaptive model-based scheme, the restriction on the system uncertainty bound being small in the case of constant model [19] can be relaxed since the MAS dynamics can be completely unknown. The MAS model parameters are updated online via adaptation whenever a new MAS state measurement is obtained. The feedback gain matrix is time-varying and can force the MAS model estimation error to converge to zero quickly. Therefore, techniques such as estimation [15] and adaptation [16] are used in this paper to derive novel distributed event-triggered consensus control scheme with unknown system dynamics. First, system states are estimated by using an adaptive state estimator or adaptive model. By using the current model parameters, the feedback gain matrix is generated once when the event triggering condition is not met. During other times, it will generate the state estimates for generating control input but the gain matrix is fixed between the events. Second, adaptive model-based event-triggered

consensus control scheme is derived based on the estimated state vector which can maintain the stability even with unknown system dynamics.

2.5.1 Adaptive Estimator

Based on event-triggered design schemes [7-14], [19], the parameters of adaptive state estimator will be updated only when the event is triggered and sensed MAS system states are received at the controller. Recalling system dynamics (1), event-triggered MAS system and adaptive state estimator with received information $x_i(t)$ can be represented as

$$\dot{x}_i(t) = f(x_i(t)) + u_i(t) + d_i(t) = \theta_i^T z_i(t),\ \forall i = 1, 2, ..., N \tag{2.13}$$

$$\dot{\hat{x}}_i(t) = \hat{f}_i(x_i(t)) + u_i(t) + \hat{d}_i(t) = \hat{\theta}_i^T(t) z_i(t) \tag{2.14}$$

where $\theta_i = [W_{fi}\, W_{di}\, 1]^T$,$\hat{\theta}_i(t) = [\hat{W}_{fi}(t)\, \hat{W}_{di}(t)\, 1]^T$ represent the target and estimated MAS dynamics of the model respectively, and $z_{i,k} = [\varphi^T(x_i(t))\ \vartheta^T(t)\ u_i^T(t)]^T$ denotes the augment state vector. Next, the MAS state estimation error dynamics $e_{si}(t)$ can be derived as

$$\dot{e}_{si}(t) = \dot{x}_i(t) - \dot{\hat{x}}_i(t) = \theta_i^T z_i(t) - \hat{\theta}_i^T(t) z_i(t) = \tilde{\theta}_i^T(t) z_i(t) \tag{2.15}$$

with $\tilde{\theta}_i(t), \forall i = 1, 2, ..., N$ is the parameter estimation error. Next, defining the update law for the unknown parameters $\hat{\theta}_i(t)$ as

$$\dot{\hat{\theta}}_i(t) = \alpha_{\theta i}\hat{\theta}_i(t) + \alpha_{ei}\gamma_i(t) z_i(t) e_{si}^T(t), \forall i = 1, 2, ..., N \tag{2.16}$$

where $\alpha_{\theta i}, \alpha_{ei}$ is the tuning parameter for $agenti$ that satisfies $0 < \alpha_{\theta i}, \alpha_{ei} < 1$ and $\gamma_i(t)$ is an indicator for the event trigger condition, i.e.

$$\gamma_i(t) = \begin{cases} 1 & \text{event is initiated} \\ 0 & \text{event is not initiated} \end{cases}, \forall i = 1, 2, ..., N \tag{2.17}$$

Meanwhile, adaptive parameter estimation error dynamics $\tilde{\theta}_i(t)$ can be expressed as

$$\dot{\tilde{\theta}}_i(t) = -\alpha_{\theta i}\tilde{\theta}_{i,k}(t) - \alpha_{ei}\gamma_i(t)z_i(t)e_{si}^T(t), \forall i = 1, 2, ..., N \tag{2.18}$$

Compared with traditional adaptive estimator schemes [15] where the updates are done periodically, event-based non-periodic tuning law is used here which tunes the state estimator in a non-periodic manner.

Remark 2: : Since consensus control gain matrix is selected using the multi agent system model parameters, the actual system (1) can be unstable. This model-based approach can work for unstable systems.

In adaptive model-based distributed event-triggered multi-agent system, the adaptive state estimator and feedback gain matrix should be updated when the above is not satisfied.

2.5.2 Adaptive model-based event triggered control for uncertain system

Compared with traditional ZOH and fixed model-based event-triggered consensus control schemes, proposed adaptive model-based consensus methodology has to consider not only the impact from event triggering, but also effects from unknown system dynamics. The consensus control input is generated by using estimated MAS states.

Consider the heterogeneous MAS and adaptive estimator represented by (1) and (11) respectively with adaptive model-based event-triggered consensus control input $u_i(t) = -\hat{f}_i(\hat{x}_i(t)) + \hat{x}_i(t) - \sum_{ij=1, i\neq j}^{N} a_{ij}\left(x_i(t) - x_j(t)\right) + \hat{d}_i(t)$ latest received $agenti$ neighbors system state. The closed loop MAS system can be represented after applying the distributed consensus control input as

$$\dot{x}_i(t) = -L_i x_i(t) - L_i e_i(k) + f_i(e_i(k)) - \hat{f}_i(\hat{x}_i(t)) + d_i(t) - \hat{d}_i(t) \tag{2.19}$$

where $\left\| f_i(x) - \tilde{f}_i(x) \right\| \leq \Delta_i$ the disturbance uncertainty.

In our proposed adaptive model-based distributed event triggered scheme, the designed distributed consensus control input $u_i(t) = -\hat{f}_i(x_i(t)) + x_i(t) - \sum_{ij=1, i\neq j}^{N} a_{ij}\left(x_i(t) - x_j(t)\right)$ which is based on adaptive MAS model $(\hat{f}_i(x_i(t), \hat{x}_i(t))$ can render the MAS estimator $\dot{\hat{x}}_i(t) = \hat{f}_i\left(x_i(t)\right) - \hat{f}_i\left(\hat{x}_i(t)\right) - \sum_{j=1}^{N} a_{ij}\left(\hat{x}_i(t) - \hat{x}_j(t)\right)$ stable asymptotically. The closed-loop adaptive model-based distributed event-triggered system is also Input-to-state stable (ISS) [8][19] with respect to the estimation error $e_i(t), \forall i = 1, 2, ..., N$.

In adaptive model-based distributed event-triggered MAS, the adaptive state estimator and feedback gain matrix should be updated when the following is not satisfied

$$\|e_i(t)\|^2 \leq \frac{\sigma_i\left(1 - l_1 N_i - l_2\right)\|z_i(t)\|^2}{\left(\frac{1}{l_1} N_i + \frac{l_i^2}{2 l_2}\right)} \tag{2.20}$$

with $0 < \sigma_i < 1$ and $l_1 < \frac{0.9}{N_i}$,$0 < l_2 < 1 - l_1 N_i$

2.6 Simulation Results

In this section, the performances of proposed two distributed event-triggered consensus control scheme (i.e. ZOH and adaptive model-based) are evaluated.

Example: Consider a 3 agent MAS as $\dot{x}_i(t) = (x_i^3(t) - x_i(t)) + u_i(t) + \mathrm{d}_i(t)$ with initial conditions $x_1 = 5, x_2 = -2, x_3 = -3$. The distributed event triggering condition parameter and connectivity graph as shown in figure 1 except the 4th agent. The disturbance $d(t)$ has upper limit of 0.00198.

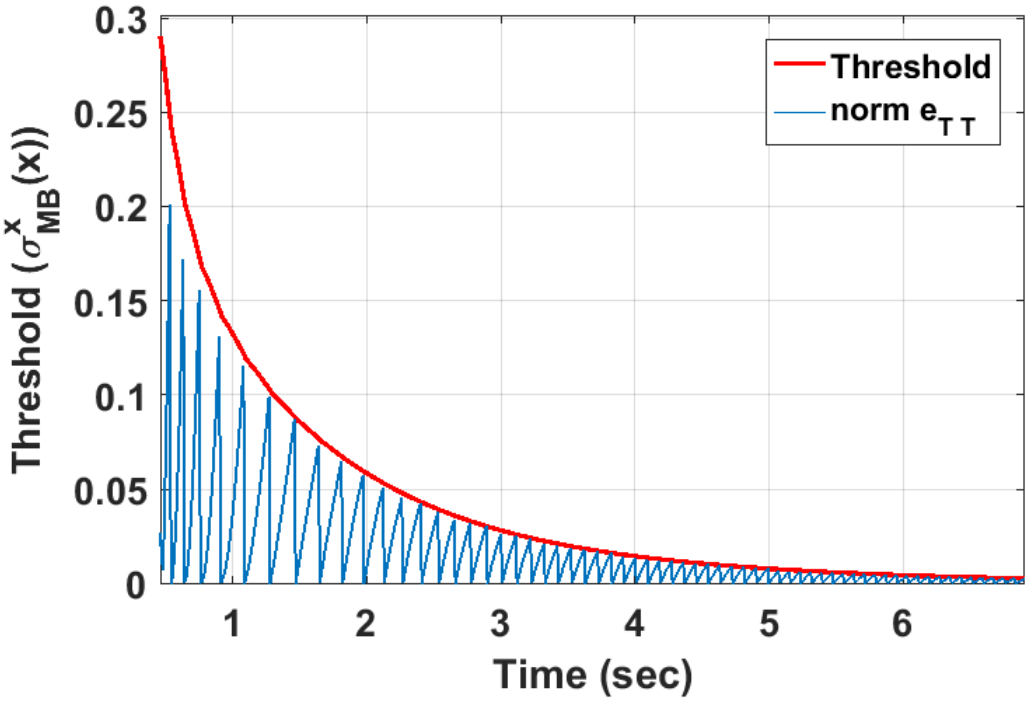

Figure 2.3: Performance of ZOH scheme error norm $\|e_k\| = \sum_{i=1}^{N} \|e_{i,k}\|$ and distributed event triggering condition

First, the performance of the ZOH distributed event-triggered consensus control scheme is evaluated. The measurement error will be reset to zero when the norm of measurement error exceeds the threshold. The sensor will transmit sensed MAS system state to the controller since event is triggered. Figure 3 shows the total measurement error (i.e. $\|e_k\| = \sum_{i=1}^{N} \|e_{i,k}\|$) and its threshold (9). In adaptive model-based event-triggered control scheme, the model estimates as initialized as zero matrices and subsequently updated on the fly. The model parameters are tuned online to attain their target values over time. The adaptive tuning parameter is selected as

$\alpha_{e1} = 0.01$,$\alpha_{e2} = 0.05$ and $[\alpha_{e3} = 0.02$. $\alpha_{\theta 1} = -2, \alpha_{\theta 2} = -0.25$ and $\alpha_{\theta 3} = -0.75$. In Figure 4, the performance of MAS adaptive model-based error events is shown.

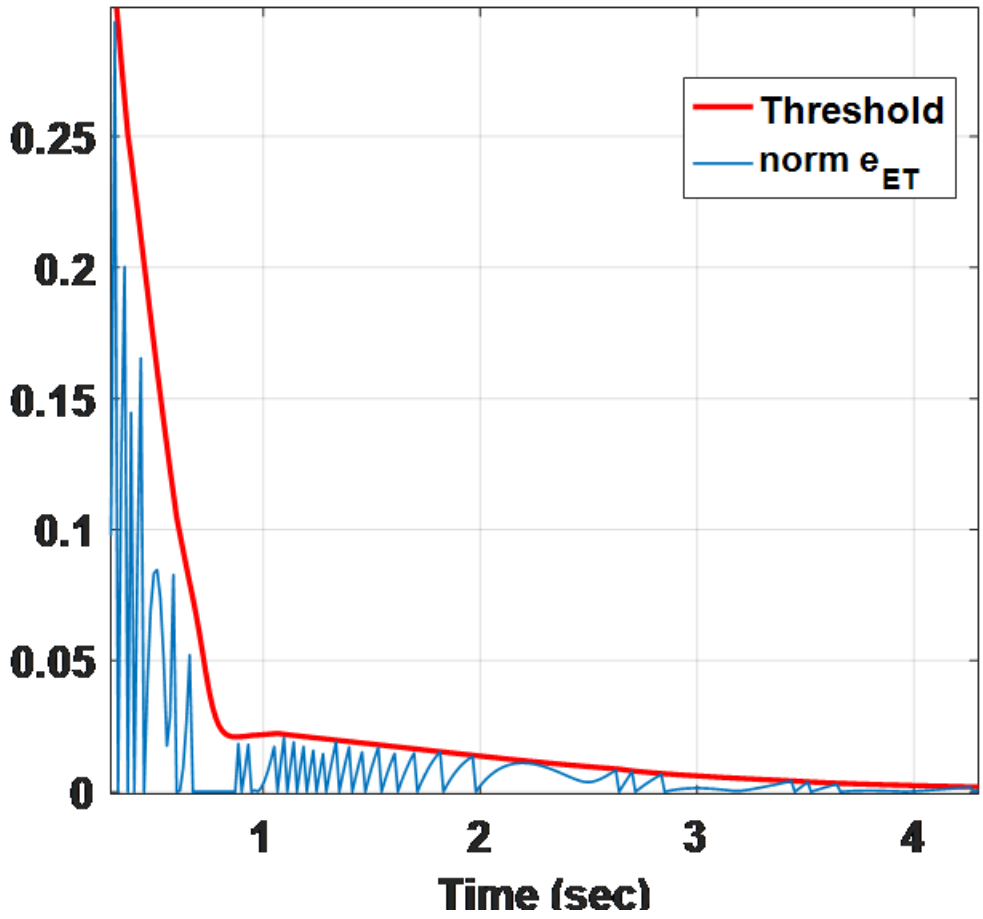

Figure 2.4: Performance of the MAS adaptive model-based error norm $\|e_k\| = \sum_{i=1}^{N} \|e_{i,k}\|$ and the distributed event triggering condition

At the beginning, the MAS estimation error is large and error events are triggered more frequently since MAS model needs to be tuned. After a short period, error events are obviously triggered much less frequently which would reduce more network traffic than ZOH distributed event-triggered consensus scheme. As can be compared from Figure 3 and Figure 4, the error rate and consensus tracking is at an advantage in adaptive model-based scheme.

2.7 Conclusion

In this paper, a novel distributed event-triggered consensus control approach consisting of non-linearity in state space model of MAS, a distributed event triggering condition was derived. Even without the system dynamics, the proposed adaptive model-based distributed event triggering scheme can not only improve the performance of control system but also saves the network resources over the ZOH and fixed model-based schemes. However, an initial online tuning phase of MAS adaptive estimator is needed before the performance improvement can be observed. Lyapunov theory demonstrates asymptotic stability and consensus.

2.8 References

1. H. T. Toivonen, "Sampled-data control of continuous-time systems with an H-infinite optimality criterion", *Automatica*, vol. 28, pp. 45-54, 1992.

2. Y. Hong, J. Hu, and L. Gao, "Tracking control for multi-agent consensus with an active leader and variable topology", *Automatica*, vol. 42, no. 7, pp. 1177-1182, 2006.

3. F. Xiao, L. Wang, J. Chen, and Y. Gao, "Finite-time formation control for multi-agent systems", *Automatica*, vol. 45, no. 11, pp. 2605-2611, 2009.

4. M. Mazo Jr., and P. Tabuada, "Decentralized event-triggered control over wireless sensor/actuator networks", *IEEE Trans. on Automat. Contr.*, vol. 56, pp. 2456-2461, 2011.

5. X. Wang, Y. Hong, J. Huang, and Z. P. Jiang, "A distributed control approach to a robust output regulation problem for multi-agent linear systems", *IEEE Trans. Automat. Contr.*, vol. 55, no. 12, pp. 2891-2895, 2010.

6. H. Zhang, and F. L. Lewis, "Adaptive cooperative tracking control of high-order nonlinear systems with unknown dynamics", *Automatica*, vol. 48, no. 7, pp. 1432-1439, 2012.

7. A. Anta, and P. Tabuada, "To sample or not to sample: self-triggered control for nonlinear system", *IEEE Trans. on Automat. Contr.*, vol. 55, pp. 2030-2042, 2010.

8. A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-triggered control for discrete-time systems", *in Proc. Amer. Contr. Conf.*, pp. 4719-4724, 2010.

9. P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks", *IEEE Trans. on Automat. Contr.*, vol. 52, pp. 1680-1685, 2007.

10. E. D. Sontag, "Input to state stability: basic concepts and results", *in Nonlinear and Optimal Control Theory*, pp. 163-220, Springer, Berlin/ Heidelberg, 2008.

11. D. V. Dimarogonas, and K. H. Johanson, "Event-triggered control for multi-agent systems", *in Proc. IEEE Conf. Decision and Contr.*, pp. 7131-7136 , 2009.

12. D. V. Dimarogonas, and K. H. Johanson, "Event-triggered cooperative control", *in Proc. Europ. Contr. Conf.*, pp. 3015-3020, 2009.

13. M. Rabi, K. H. Johanson, and M. Johanson, "Optimal stopping for event-triggered sensing and actuation", in Proc. IEEE Conf. Decision and Contr., pp. 3607-3612, 2008.

14. G. Pekir, and A. Shiryaev, Optimal stopping and free-boundary problems, Lectures in Mathematica, Birkhauser Verlag Press, Basel, Switzerland, 2006.

15. G. Luders, and K. Narendra, "An adaptive observer and identifier for a linear system", *IEEE Trans. on Automat. Contr.*, vol. 18, pp. 496-499, 1973.

16. P. J. Werbos, "A menu of designs for reinforcement learning over time", *J. Neural Networks Contr.*, vol. 3, pp. 835-846, 1983.

17. Z. P. Jiang, and Y. Wang, "Input-to-state stability for discrete-time nonlinear system", *Automatica*, vol. 37, pp. 857-869, 2001.

18. S. Jagannathan, Neural network of nonlinear discrete-time system, CRC Press, 2006.

19. E. Garcia, and P. J. Antsaklis, "Model-based event-triggered control with time-varying network delays", *in Proc. IEEE Conf. Decision and Contr.*, pp. 1650-1655, 2011.

20. M.S. Branicky, M.M. Curtiss, J. Levine and S. Morgan, "Sampling-based planning, control and verification of hybrid systems", *IEEE proceedings. Control theory and applications*, vol. 153-5, pp. 575-590, 2006.

21. R.W. Brennan ; M. Fletcher ; D.H. Norrie, "An agent-based approach to reconfiguration of real-time distributed control systems", *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, 2002.

22. J.H. Sandee, W.P.M.H. Heemels, P.P.J. v.d. Bosch, "Case studies in event-driven control", 2009.

23. A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-triggered control for discrete-time systems", *in Proc. Amer. Contr. Conf.*, pp. 4719-4724, 2010.

24. G.S. Seyboth, D.V. Dimarogonas, K.H. Johansson, "Event-based broadcasting for multi-agent average consensus", Automatica 49 (1) (2013) 245-252.

25. Garcia, E., Cao, Y., Wang, X., and Casbeer, D.W, "Cooperative control with general linear dynamics and limited communication: Periodic updates", *In Proc. IEEE ACC*, 3195-3200.

26. Huaipin Zhang, Dong Yue, Xiuxia Yin, Songlin Hu, Chun xia Dou, "Finite-time distributed event-triggered consensus control for multi-agent systems", *Elsevier Information Sciences*, vol 339, 20. pp. 132-142. 2016.

27. A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-triggered control for discrete-time systems", *in Proc. Amer. Contr. Conf.*, pp. 4719-4724, 2010.

28. P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks", *IEEE Trans. on Automat. Contr.*, vol. 52, pp. 1680-1685, 2007.

29. H. Su, G. Chen, X. Wang, and Z. Lin, "Adaptive second-order consensus of networked mobile agents with nonlinear dynamics", *Automatica*, vol. 47, no. 2, pp. 368-375, 2011.

30. Z.-G. Hou, L. Cheng, and M. Tan, "Decentralized robust adaptive control or the multiagent system consensus problem using neural networks", *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 3, pp. 636-647, Jun. 2009.

Chapter 3

Optimal Self-Triggered Control and Network Co-design for Networked Multi-Agent System via Adaptive Dynamic Programming

3.1 Abstract

In this paper, optimal self-triggered control and network co-design problem has been studied for networked multi-agent system (MAS) with uncertain physical and network dynamics. There are two main challenges, i.e. 1) effective modeling of control and network jointly is lacking, 2) uncertain physical and network dynamics complicate the optimal design significantly. To overcome these challenges, a novel control and network joint modelling algorithm has been developed for networked MAS firstly. Specially, we consider the self-triggered control design from physical system aspect and admission control design from network

aspect. Next, adopting adaptive dynamic programming strategy, a novel online time-based optimal self-triggered control and network admission co-modelling has been proposed. The developed scheme could not only optimize both self-triggering time and number of admitted agents, but also relaxes the requirement of networked MAS physical system and network dynamics. For results we include simulation results as well.

Keywords - Networked multi-agent systems, admission control, self-triggered control, consensus, Optimal co-design.

3.2 Introduction

Past decade has witnessed a significant increase in research of networked multi-agent systems (MAS) and its implications for real-time implementations. Researchers from controls as well as network and computation are invested in how to optimally control both physical and network systems. Traditionally, networked MAS is described as a feedback MAS control system. Here the control parameters and control loops are closed through a shared communication network. Since information (e.g. sensor data, actuator data, reference line, target line, etc.) is exchanged through a network connected MAS control system components (controllers, actuators, sensors, etc.,) in networked MAS [2], network controllers could share information in a more efficient manner and further improve networked MAS performance from control and network aspects simultaneously. However, before reaping these advantages, there are several challenges that are to be solved, i.e. 1) how to model the networked MAS from physical control system and practical network efficiently, 2) how to optimize the physical system performance as well as upgrade the effectiveness of network resource consumption.

Recently, authors in [3] model network model based random delays and packet dropouts and further investigate how network imperfections, i.e. delays and packet dropouts, affect the stability of networked control systems (NCS). Instead of stability, optimality is much more preferred. Therefore, [4] has considered the optimal design problem for NCS with network imperfections. Specifically, network-induced delays are considered as random variables. Moreover, adopting stochastic optimal theory, an infinite horizon stochastic optimal control can be developed for Networked Control System.

However, most existing NCS literatures, e.g. [3], [4], have several issues that needs to be addressed: 1) The relationship between network imperfections and network protocol has not been investigated effectively. 2) The consideration about networked MAS is lacking. 3) Real-time optimal network and control co-design for networked MAS is extremely needed 4) in literature for NCS needs the knowledge of system dynamics which cannot be known beforehand.

Next, considering the multi-agent consensus control problem, authors in [5] presented event-triggered consensus control where a discrete time based event driven scenario has been proposed, the information exchange happens only when required. Furthermore, in [6], authors developed a discrete time based scheme that controller does not have to keep track of the errors all the time. Authors demonstrated that self-triggered control is less demanding from the real-time controllers. However, [5] and [6] assumed that the network communication will not be contaminated by latency. This is not a practical case. In literatures like [7] and [8], authors presented network models with delays. But it fails to truly integrate the physical system and network channel as one entire control system. In this paper, we will develop a novel control scheme where availability of real-time network resources is considered before admitting or dis-admitting a new physical agent to transmit over the

communication channel. This scheme is then integrated with the physical control systems and shared network to form a novel co-design that can benefit the network and MAS simultaneously.

The main contributions of this paper includes 1) A novel co-model of networked multi-agent consensus control; 2) A novel optimal co-design has been proposed including an admission control scheme from network aspect that allows new agents to be admitted into network based on network usage history and physical system requirement and an optimal control scheme; and 3) Consensus stability is guaranteed.

This paper is organized as: Section-2.3 problem formulation for optimal networked MAS co-design. Section-2.4 where admission control scheme and distributed self-triggered scheme with infinite horizon optimal consensus control is discussed. And Section 2.5 demonstrates the effectiveness of the proposed scheme by simulation results and 2.6 concludes the paper.

3.3 Problem Formulation

3.3.1 Networked Multi-Agent Systems (MAS)Co-Modeling

As shown in figure 1, wireless channel closes the loop for the control system. The main objective for co-modeling is to effectively integrate the MAS physical system aspect with network aspect into a dynamic model framework. In this paper, the co-design for admission control from network aspect and MAS control from physical system aspect has been considered. Therefore, the key point is to integrate the admission control aspect with physical system dynamics. Inspired from [22], a

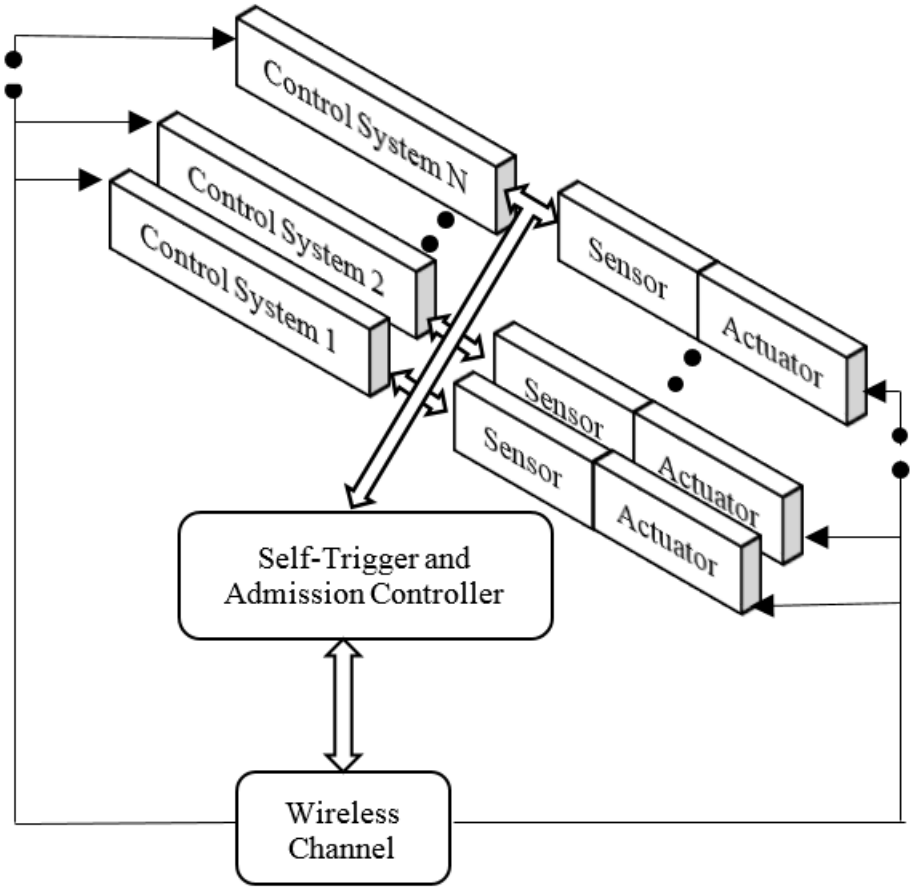

Figure 3.1: Networked Control System

novel networked MAS co-design can be represented as

$$x_{i,k+1} = f_i(x_{i,k}, \tau_{i,k}) + g_i(x_{i,k}, \tau_{i,k})u_{i,k} \tag{3.1}$$

where $\tau_{i,k}$ is the network delay of agent i at time instant k caused by network imperfections, $x_{i,k}, u_{i,k}$ denote states and control input of agent i at time instant k respectively. It is important to note that the co-model (1) is generated from both admission control (i.e. network aspect) and MAS consensus control (i.e. physical system aspect). The details are shown as follows

Admission Control (Network Aspect): As in [22], the network imperfection in-

duced by network can be represented as

$$\tau_{i,k+1} = \frac{L}{\frac{B_W}{Q_L}\log_2\left(N + \frac{1}{\mu_{i,N}}\right)} \tag{3.2}$$

with $\tau_{i,k} \in \Re$ denoting the delay induced by admission control,$L \in \Re$ as the packet length in buffer queue Q_L, $B_W \in \Re$ as the maximum bandwidth available, N being the number of agents allowed at a particular instant when B_W is the corresponding bandwidth and $\mu_N = \frac{Q_L L}{B_w \log_2(N)}$. The theory of admission controller is given in section 2.4.

Consensus Control (MAS physical system Aspect):

$$x_{i,k+1} = f_i(x_{i,k}, \tau_{i,k}) + g_i(x_{i,k}, \tau_{i,k}) u_{i,k+\tau^i{}_k} \tag{3.3}$$

where $x_{i,k} \in \Re^n$ is the state of the agent i at time k, $u_{i,k+\tau^i{}_k} \in \Re^n$ is input to system for agent i at time k with delay (2) τ_k . And f, g are non-linear functions of the control system dynamics.

Next, we formulate the infinite horizon optimal co-design problem.

3.3.2 Infinite Horizon Optimal Co-design

Inspired from [21], we need to minimize the cost function defined below by the infinite horizon optimal co-design method

$$V_i\left(x_i, t\right) = \phi_i(x_i(t_\infty)) + \left(\int_t^\infty r_i\left(x_i(s), u_i(s)\right) ds\right) \tag{3.4}$$

with $r_i(x_i, u_i) = x_i^T(t) M_i x_i(t) + u_i^T(t) R_i u_i(t)$, being cost-to-go, $\phi_i(x_i(t_\infty)) = x_i^T(t_\infty)$ $S_i(t_\infty) x_i(t_\infty)$, with $S_i(t_\infty) \in \Re^{n \times n}$ denoting terminal weighing matrix [23], $M_i \in$

$\Re^{n\times n}$, $R_i \in \Re^{m\times m}$ represents the symmetric positive semi-definite matrix and the positive definite matrix respectively while t_∞ represents the time far away in future. In addition, the optimal value function is defined as in [5]. For sake of simplicity we consider $f_i(x) = A_i x_i$, and $g_i(x) = B_i$ where matrices $A \in \Re^{n\times n}$ and $B \in \Re^{n\times m}$. Therefore, using [25] and (4), we have

$$V_i^*(x,t) = \min_{u_i} \left[x_i^T(t_\infty) S_i(t_\infty) x_i(t_\infty) + \left(\int_t^\infty \left(x^T M_i x_i + u_i^T R_i \, u_i \right) ds \right) \right] \tag{3.5}$$

According to Bellmans principal of optimality [21], the optimal cost function for the first part needs to satisfy the following Bellman Equation as

$$\begin{aligned} V_i(x,t) &= \left(\int_t^\infty r_i \left(x_i(t), u_i(t) \right) + V_i(x_i, t_\infty) \right) \\ V_i(x_i, t_\infty) &= \phi_i(x_i(t_\infty)) \end{aligned} \tag{3.6}$$

Here $V_i(x_i, t_\infty) = \phi_i(x_i(t_\infty)) = x_i^T S_i(t) x_i$ with $S_i(t) \in \Re^{n\times n}$ being the solution of Recatti Equation defined as

$$A_i^T S_i(t) + S_i(t) A_i - S_i(t) B_i R_i^{-1} B_i^T S_i(t) + M_i + \dot{S}_i(t) = 0 \tag{3.7}$$

where $\dot{S}_i(t)$ is the derivative of $S_i(t)$ with respect to time t. A_i and B_i are assumed as stated earlier for agent i. The optimal input is obtained using optimal control theory [21], we can develop constrained optimal co-design as

$$u_i(t) = -K_i^* x_i(t) = -R_i^{-1} B_i^T S_i(t) x_i(t) \tag{3.8}$$

where Kalman gain K_i^* is given by $K_i^* = R_i^{-1} B_i^* S_i(t)$.

Remark 1: To obtain the optimality condition, the Ricatti Equation is sufficient

and to solve it requires the model to be known in full detail. Therefore, solving the ricatti Equation in a forward manner of time with unknown system model is a difficult task.

3.4 Admission Control Scheme and Self-Triggered Consensus Control for Networked MAS

3.4.1 Admission Control Scheme

Consider buffer dynamics of a wireless network system [12] given by the form,

$$\hbar_i(k+1) = \chi_i\left(\eta_i\left(\hbar_i(k)\right) + T_s\mu_{i,s}(k)\right) \tag{3.9}$$

where $\hbar_i(k) \in \Re^n$ buffer occupancy at the k-th time instant i, T_s interval of measurement, $\mu_{i,s}(k) \in \Re^n$ is the source rate that is defined by feedback for agent i, $\eta_i\left(\hbar_i(k)\right)$ is a non-linear function of buffer occupancy, source rate and service capacity, $\chi_i(\bullet)$ is a saturation function that satisfies the following criteria:

$$\begin{aligned}\chi_i(c) &= 0, \;\; if \;\; c \leq 0,\\ &= i, \;\; if \;\; c \geq i,\\ &= c, \;\; otherwise\end{aligned}$$

For implementation purposes we will ignore the saturation function. And go ahead with the following updated function:

$$\hbar_i(k+1) = \eta_i\left(\hbar_i(k)\right) + T_s\mu_{i,s}(k) \tag{3.10}$$

We define the instantaneous bandwidth requirement using the traffic that is being accrued at each measurement interval and is known. The packet/cell loss

requirement (PCR) is also used here.

$$\Delta Bw_i(k) \;=\; \frac{\eta_i\left(\hbar_i(k)\right) - \eta_i(\hbar_i(k-1))}{T_s} \tag{3.11}$$

where$\Delta Bw_i(k)$ is the additional bandwidth that is required by agent i, with T_s as the sampling time. Further, the bandwidth requirement is given by:

$$Bw_i(k+1) = Bw_i(k) + \Delta Bw_i(k) \tag{3.12}$$

where $Bw_i(k+1) \;\; and \;\; Bw_i(k)$ represents the bandwidth at the time instant $k+1 \;\; and \;\; k$ for agent i, respectively. We always make sure the bandwidth requirement does not exceed a maximum value. We denote it as $S_{\max}$.

The available capacity is calculated by subtracting the current bandwidth usage from all existing sources from maximum available bandwidth of the physical link.

$$Available\_capacity \;=\; S_{\max} - \sum_{i=1}^{n} Bw_i(k) \tag{3.13}$$

Using available buffer space as an input to the controller, available network resources can be calculated as:

$$Available\_network\_resources = \Phi_{\max} - \sum_{i=1}^{n} \hbar_i(k) \tag{3.14}$$

where $\Phi_{\max}$ is the max buffer for the switch/network router. Queue length $q_i\,(k)$, and its past values are used to generate congestion indicator, and round-trip (RRT) delays at the communicating node. The congestion indicator flag is set when the past several buffer occupancy values are about 90% full, the rate of change of the queue length is positive and high, the round-trip delays are large.

The algorithm can be summed up using following pseudo code.

Algorithm 1: Pseudo Code for Estimation of Network Congestion

Result: Congestion Flag Set or Reset

if $q_i\,(k) > 90\%$ ***and*** $q_i\,(k-1) > 90\%$ ***and*** $(RRT > 2RRT_{min})$ **then**
    Congestion flag = *True*;
else
    Congestion flag = *False*;
end

With the correct estimation of bandwidth we check for admittance. Following algorithm sums up the admittance pseudo code.

Algorithm 2: Pseudo Code for Agent Admittance

Result: Admit or Reject Source

if $available\ network\ resources > 10\%$ ***and*** $congestion\ flag == True$ ***and*** $available\ capacity > PCR$ **then**
    admit source *"i"*;
else
    reject;
end

3.4.2 Infinite Horizon Optimal Distributed Self-Triggered Consensus Control

Using (2) and admission control of previous subsection, we calculate the delays that are being experienced by the network. For simplicity of derivation we will assimilate the delay τ (2) in time $t + \tau$ and denote it simply as t.

Next, we consider N agents, with states denoted by x_i for agent i. Further the dynamics of the agents are supposed to obey single integrator model as:

$$\dot{x}_i = u_i \qquad i \in \{1, 2, ..., N\} \tag{3.15}$$

where u_i signifies control input to the system.

The agreement control law in [20] is given by

$$u_i = -K_i \sum_{j \in N_i} (x_i - x_j) \tag{3.16}$$

As shown in [20] for a connected graph, all agents states converge to a common agreement point which is average of the initial states of the agents

$$\frac{1}{N} \sum_{i \in N} x_i(0) \tag{3.17}$$

Next, a model-free reinforcement learning based algorithm will be developed to obtain the optimal consensus control.

Recall to the Bellman equation [2], for a continuous time system, the Q-function

can be expressed as

$$\begin{aligned} Q_i(x_i,u_i,t) &= V_i(x_i,t) = x_i^T S_i(t) x_i \\ &= \begin{bmatrix} x_i \\ u_i \end{bmatrix}^T \begin{bmatrix} \dot{S}_i(t) + A_i S_i(t) + S_i(t) A_i + Q_i + S_i(t) & S_i(t) B_i \\ B_i^T S_i(t) & R_i \end{bmatrix} \begin{bmatrix} x_i \\ u_i \end{bmatrix} \\ &= \begin{bmatrix} x_i \\ u_i \end{bmatrix}^T \begin{bmatrix} G_{i,xx}(t) & G_{i,xu}(t) \\ G_{i,ux}(t) & G_{i,uu}(t) \end{bmatrix} \begin{bmatrix} x_i \\ u_i \end{bmatrix} \end{aligned} \tag{3.18}$$

with G-matrix represented by

$$G_i(t) = \begin{bmatrix} G_{i,xx}(t) & G_{i,xu}(t) \\ G_{i,u}(t) & G_{i,uu}(t) \end{bmatrix}, t \in [t,\infty), G_i(t_\infty) = \begin{bmatrix} S_i(t_\infty) & 0 \\ 0 & 0 \end{bmatrix}, t = t_\infty \tag{3.19}$$

The optimal control gain ${K_i}^*$ can be defined as follows using [23] and (18),

$${K_i}^* = R^{-1} B^T S_i(t) = G_{i,uu}^{-1}(t) G_{i,ux}(t) \tag{3.20}$$

Similar to [23], to learn the Q-function and G-matrix a model-free online tuning is incorporated.

Using the result of [23], continuous time-varying action which is uniform in nature, the dependent function [24], $Q_i(x_i,u_i,t)$, the vector form can be expressed as

$$Q_i(x_i,u_i,t) = {z_i}^T(t) G_i(t) z_i(t) = {\theta_i}^T \sigma_i(t) \bar{z}_i(t) = {\theta_i}^T \psi_i(z_i,t) \tag{3.21}$$

where ${\theta_i}^T \sigma_i(t) = \bar{g}_i(t) = vec^T(G_i(t)) \in \Re^{l \times \frac{(n+m)^2}{2}}$ with $\theta_i \in \Re^p$ is the parameter vector which is the target and $\sigma_i(t) \in \Re^{p \times \frac{(n+m)^2}{2}}$ represents the basis function which is time dependent, of the Q-function estimator. Moreover, the argument state $z_i(t)$ is defined as $z_i(t) = \begin{bmatrix} {x_i}^T(t) & {u_i}^T \end{bmatrix}^T \in \Re^{n+m=l}$ and

$\bar{z}_i(t) = [z_{i,1}^2(t), ..., z_{i,1}(t)z_{i,l}(t), z_{i,2}^2(t), ..., z_{i,l-1}(t)z_{i,l}(t),$
$z_{i,l}^2(t)]$ is the Kronecker product quadratic polynomial basis vector. In addition, $vec()$ function is represented alike [25]. Figure 3 enumerates the steps needed to be followed in the form of flowchart. The function estimator parameter $\theta_i(t)$ will be updated at each time instant by using (22). We force the terminal cost estimation error and temporal difference (TD) error to zero and then we update the control gain by using (20) and tuned Q-function parameter, we define

$$\hat{\theta}_i(t + \Delta t) = \hat{\theta}_i(t) + \alpha_{i,\theta} \frac{\Delta\psi_i(z_i, t) {e_i}^T(t)}{\Delta {\psi_i}^T(z_i, t)\Delta\psi_i(z_i, t) + 1} + \alpha_{i,\theta} \frac{\sigma_i(t_\infty)\bar{z}_i(t)e_{i,z}^T(t)}{\|\sigma_i(t_\infty)\bar{z}_i(t)\|^2 + 1} \tag{3.22}$$

where $\alpha_{i,\theta}$ is tuning parameter, $0 < \alpha_{i,\theta} < 1$,
$e_i(t) = -\hat{\theta}_i^T(t)\Delta\psi_i(z_i, t) + \int_t^{t+t_s} r(x_i(\tau), u_i(\tau))d\tau$, and

$$e_{i,z}(t) = \phi_i(x_i(t_\infty)) - \hat{\theta}_i^T(t)\sigma_i(t_\infty)\bar{z}_i(t).$$

For proof refer to [23]. When we reach the final time, the proposed approach is supposed to stop . If not then the estimated control and the Q-function estimator gain is supposed to be tuned at the next sampling instant. Next, to better improve the efficiency of the network resource, the self-triggering technique has been integrated with the optimal consensus control and further develop the distributed self-triggered optimal consensus control for networked MAS. The distributed self-triggering condition has been derived as follows. The distributed-self-triggering algorithm leverages previous work [6] and [7].

Figure 2 shows the time line of control inputs for agents j and i. Observe here that the control inputs for different agents are given different notations. However, in the proposed algorithm, the control inputs are triggered for all the agents even

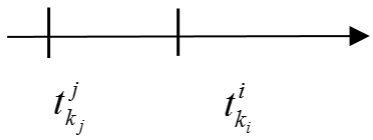

Figure 3.2: Time line of control inputs for agents j and i

though the trigger condition was met by only one of the agents. So we have

$$u_i(t^i{}_k) = u_j(t^i{}_k) \quad \forall i, j = 1, 2, ..., N \tag{3.23}$$

The algorithm shown in Figure 3 relies on present information from other agents to calculate its control over its next interval.

The control for agent i is based on the state error between vehicle i and its neighbors defined in Laplacian L matrix [7]. In deriving the optimal control for each agent we will use the optimal control gain, K_i^* which was estimated using the infinite horizon approach discussed earlier. Since the consensus is reached when the error becomes zero for all the agents in their neighborhood, the resulting control law for agent i is written as

$$u_i(t) = -K_i{}^* \sum_{j \in N_i} \left(x_i(t^i_{k_i}) - x_j(t^j_{k_j}) \right) = -K_i{}^* L_i x \tag{3.24}$$

where L_i is the i-th row of matrix and $x_i(t^i_{k_i})$ is the presently known information of agent i.

Using integral form, we calculate the state of agent j as follows,

$$x_j(t^i_k) = \dot{x}_j(t^i_k)(t - t^i_k) + x_j(t^i_k) \tag{3.25}$$

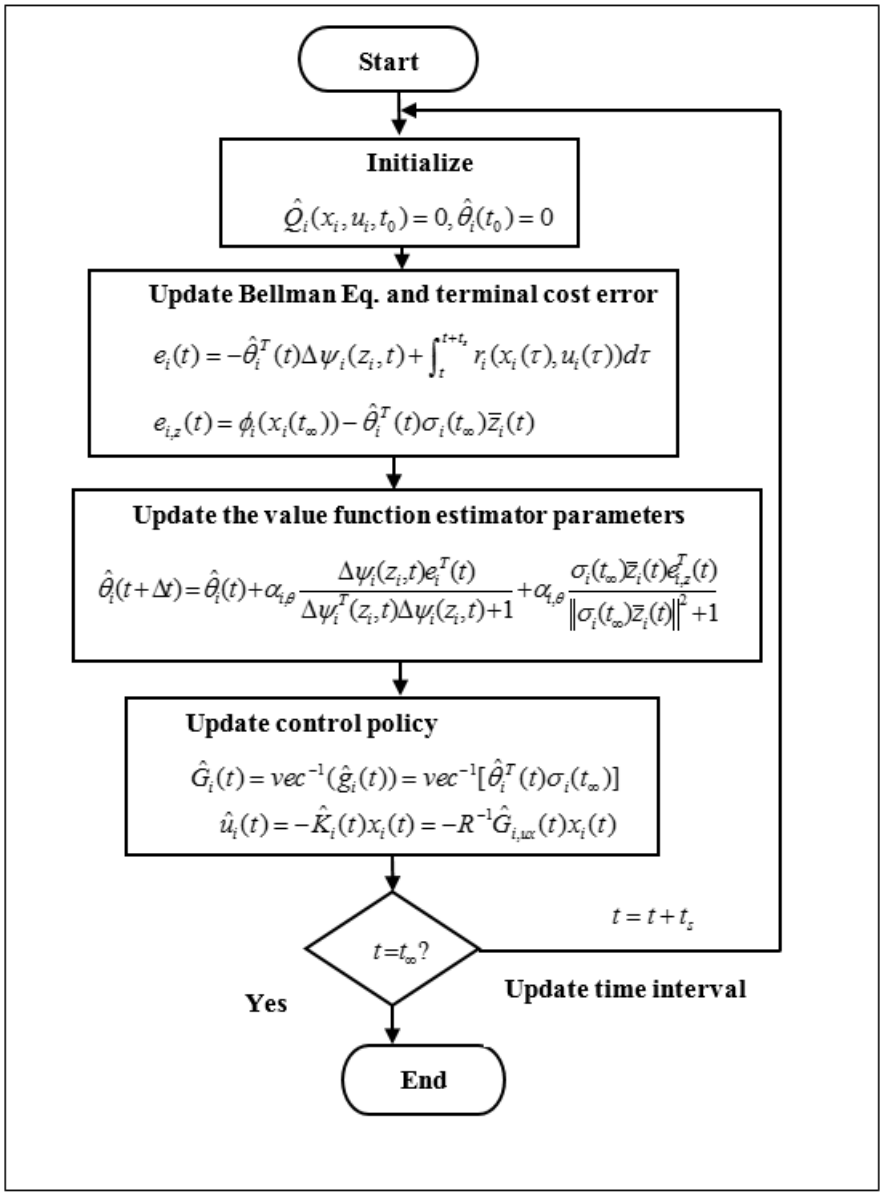

Figure 3.3: Flow chart for Infinite horizon optimal regulator

Equation (25) can be substituted in (24) and the control becomes

$$u_i(t) = -K_i^* \sum_{j\in N_i} \left(x_i(t_k^i) - \left[\dot{x}_j(t_k^i)(t-t_k^i) + x_j(t_k^i)\right]\right) \tag{3.26}$$

Since $u_i(t) = \dot{x}_i(t)$, we have

$$\dot{x}_i(t) = -\sum_{j \in N_i} \left(x_i(t^i_{k_i}) - x_j(t^j_{k_i}) \right) \tag{3.27}$$

Now, we define error of states as difference between last known state to present state.

$$e_i(t) = x_i(t^i_{k_i}) - x_i(t) \quad and \quad e_j(t) = x_j(t^j_{k_j}) - x_j(t) \tag{3.28}$$

This gives us,

$$x_i(t^i_{k_i}) = e_i(t) + x_i(t) \quad and \quad x_j(t^j_{k_j}) = x_j(t) + e_j(t) \tag{3.29}$$

Substitute all the values in (27)

$$\dot{x}_i(t) = -\sum_{j \in N_i} \left(\left(x_i(t) - x_j(t) + e_i(t) - e_j(t) - \dot{x}_j(t^i_k)(t - t^i_k) \right) \right)$$

Next, using a simplified equation with previously mentioned substitutions, we have

$$\dot{x}_i = -L_i x_i - L_i e_i + \lambda_i \tag{3.30}$$

where $\lambda_j = -\sum_{j \in N_i} (\dot{x}_j(t^i_k)(t - t^i_k))$.

Note as stated earlier, the time t considered for derivation of (29) considers the co-model parameter delay τ assimilated in itself and the optimal control gain matrix K^*_i, thus completing a truly co-modelled Network MAS.

Theorem 1 *(Distributed Self-Triggering Condition for multi agent system consensus):* In distributed self-triggered multi agent system (1), the control input should be triggered and controller should be updated when the following is satisfied

$$\left|\varsigma_i(t-t^i_{k_i})\right|^2 \leq \phi(P_i(t-t^i_{k_i})+\nu_i)^2 - \frac{\sigma_i}{2\left|N_i\right|}\sum_{j\in N_i}\frac{1}{2a}{\lambda_j}^2 \tag{3.31}$$

where $0 < \sigma_i < 1$, $\phi \triangleq \zeta_i a\left(\frac{1-\frac{3}{2}a|N_i|}{|N_i|}\right)$, $0 < a < 1$, $P_i = -\left|N_i\right|\zeta_i + \sum_{j\in N_i}\zeta_j$, $\nu_i = \sum_{j\in N_i}\zeta_j(t^i_{k_i}-t^j_{k_j}) + \left|N_i\right| x_i(t^i_{k_i}) - \sum_{j\in N_i} x_j(t^j_{k_j})\ \zeta_i = -\sum_{j\in N_i}\left(x_i(t^i_{k_i}) - x_j(t^j_{k_i})\right)$ and $\lambda_j = -\sum_{j\in N_i}(\dot{x}_j(t^i_k)(t-t^i_k))$.

Remark 3: In self-triggered consensus control approach, it is not necessary to monitor the error states continuously as in event-triggered consensus control. The next update is calculated by using present available information. This way the micro-controller has lesser overheads and self-triggered consensus control is more feasible.

3.5 Simulation Results

In this section, we present simulation results from the admission controller Multi-Agent Networked control system with Self-triggered control.

Example: For the admission controller we define network resources to be used by limited agents. Maximum agents to be allowed is confined till 6. Max buffer size is 10 bits. Max bandwidth is 90bps (bits per second) with each agent considered to be sending a variable packet length of max 30 bits. The packet length changes with the amount of control input needed to drive a particular agent. This allows for only 3 agents to transmit information at a time when states are too far apart and all 6 agents can transmit if states are not far apart. We use (30) to calculate the error and assign a linear relationship between packet length and error such that $\max(error) = \max(length) = \min(agents)$ and $\min(error) = \min(length) = \max(agents)$. Sampling time T is 0.001 sec. For the self-triggered control of MAS,

we have initial conditions of agents as,

$$x\_init \;=\; [2.35,\; 4.5,\; \text{-}3.75,\; 1.60,\; -0.35,\; \text{-}5.05]$$

$\sigma = 0.95, \quad a = 0.75$, ∂= 6, $B = I^{6\times 6}$ and

$$A = \begin{bmatrix} 1.38 & -0.2077 & 6.715 & -5.676 & 1.067 & 1.067 \\ -0.5814 & -4.29 & 0 & 0.675 & 1.343 & 5.893 \\ 1.067 & 4.273 & -6.654 & 5.893 & -4.29 & 0 \\ 0.048 & 4.273 & 1.343 & -2.104 & 4.273 & 1.343 \\ -4.29 & 1.067 & 1.067 & 0.048 & 0 & -4.29 \\ 4.235 & 0.048 & 6.715 & 4.273 & -6.654 & 0.048 \end{bmatrix}$$

with $x = [x_1^T x_2^T x_3^T x_1^T x_2^T x_3^T]^T \in \Re^{6\times 1}$. Define the performance index as $V_i\left(x_i, t_0\right) = \left[{x_i}^T(t_\infty) S_i(t_\infty) x_i(t_\infty) + \left(\int_{t_0}^{\infty} \left({x_i}^T M x_i + {u_i}^T R\, u_i \right) ds \right)\right]$ with M, R selected as $M = 5I^{6\times 6}$ and $R = I^{4\times 4}$ where I denotes identity matrix and $t_\infty = 6\,\mathrm{sec}$ and terminal weighing matrix is given as $S_i(t_\infty) = 5I^{6\times 6}$.

Next, we consider a continuous-time system. We implement our proposed scheme in this system. The state dynamics dictated by (3), in the presence of complete unknown system dynamics. Also the augmented state vector $z_i(t)$ is represented as $z_i(t) = \left[{x_i}^T(t) \quad {u_i}^T(t)\right]^T \in \Re^{12\times 1}$. An initial admissible control is chosen as $u_0 = -0.48 I^{6\times 6} x$. The regression function has state-dependent part for Q-function which is selected as $\{z_{i,1}^2, z_{i,1}z_{i,2}, z_{i,1}z_{i,3}, ..., z_{i,12}^2\}$. The tuning parameter is selected as $\alpha_{i,\theta} = 0.95$, and the initial parameters are in random range $\hat{\theta}_0 \in \begin{bmatrix} 0 & 1 \end{bmatrix}$.

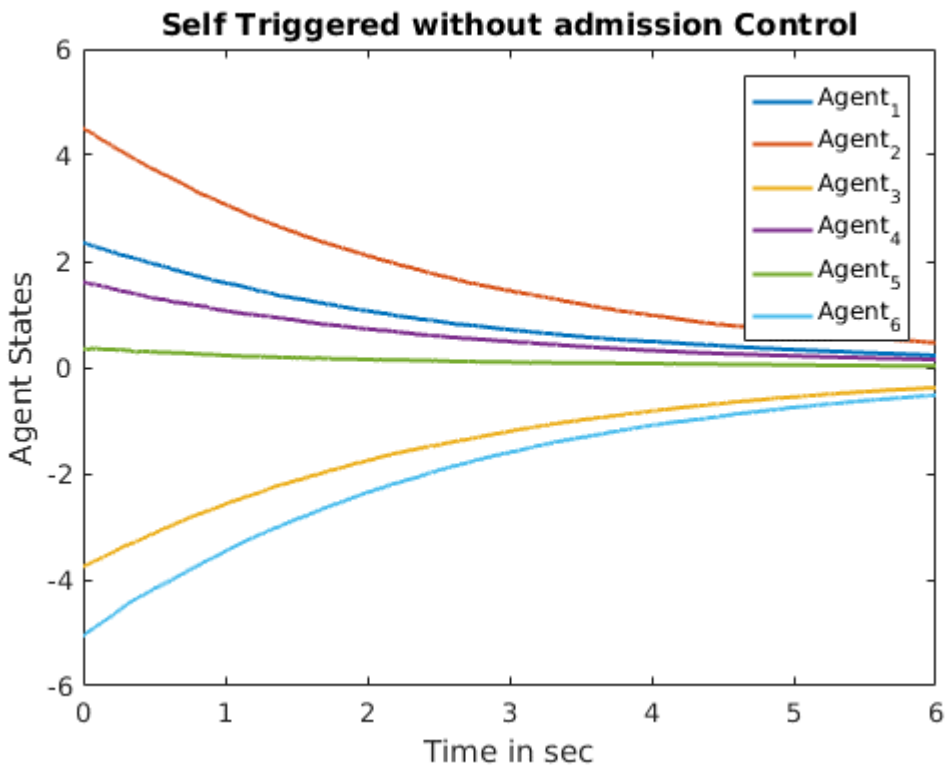

Figure 3.4: Consensus control for 6 agents with no admission control

Figure 4. shows the simulation results when the admission controller is not employed and network parameters are kept as discussed at the starting of this section. It can be seen that the consensus was not achieved in 6 seconds

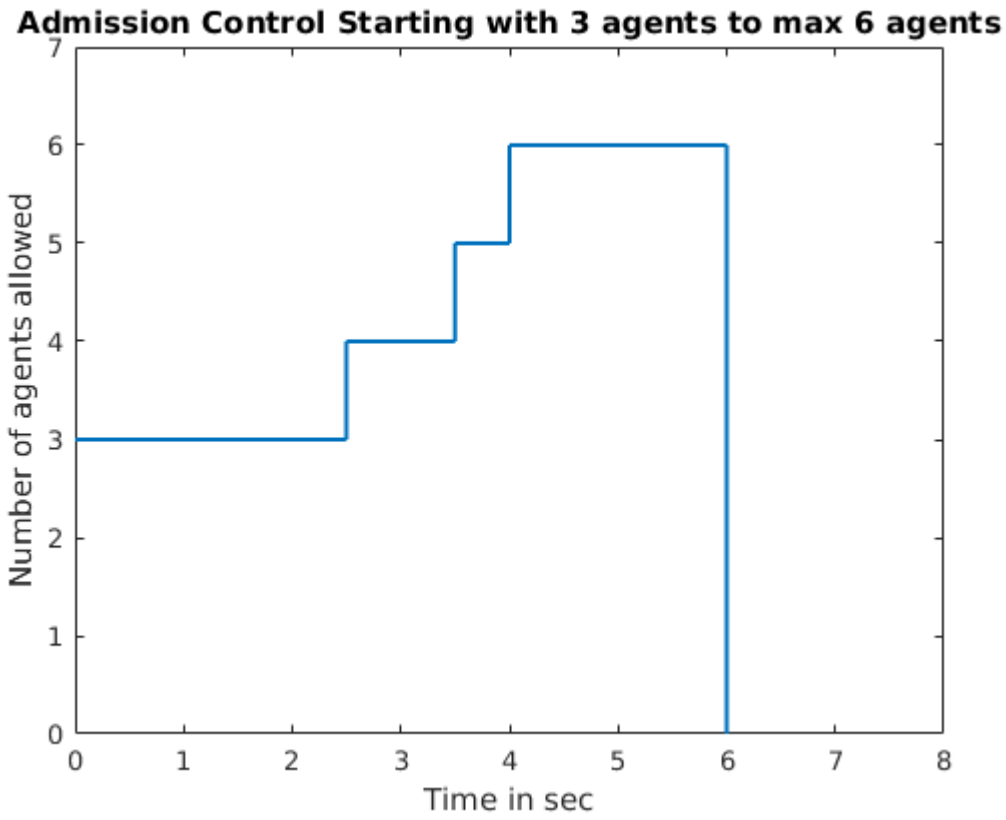

Figure 3.5: Admission control simulation

Figure 5. shows the admission controlled by NCS. Here the simulation starts with 3 agents and according to the stability of the physical system admission controller goes on adding more agents as needed.

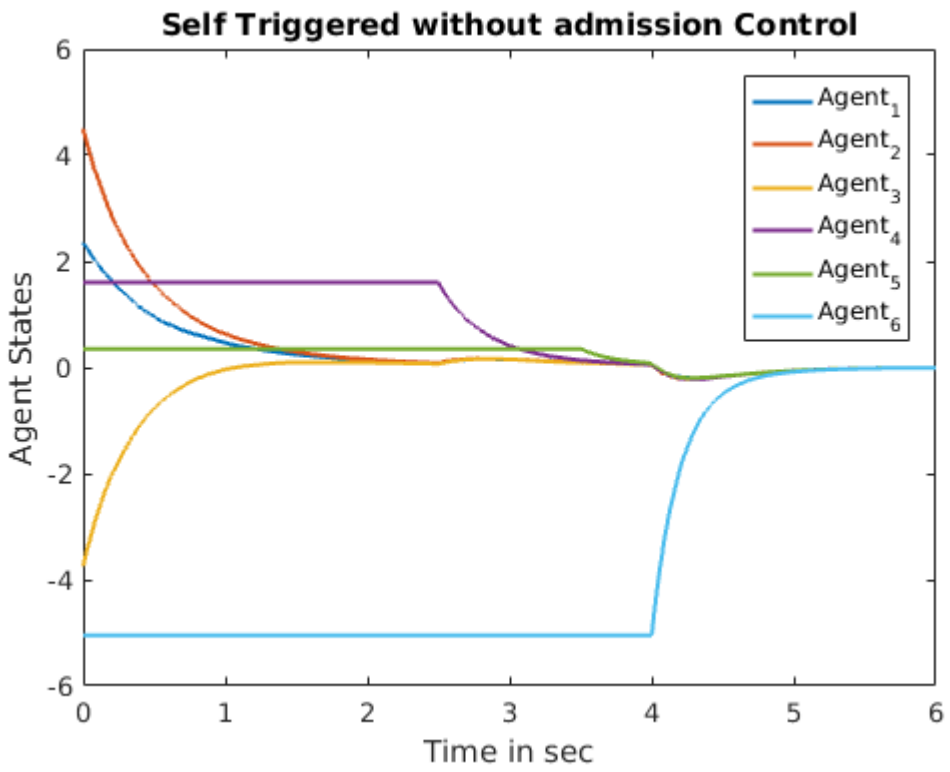

Figure 3.6: Simulation Result for Self-Triggered Control under Admission Controlled NCS

In figure 6, the self-triggered control simulation starts with first 3 agents with initial states at 2.35, 4.5 and -3.75. When the agents start to converge and the system is in relatively stable state admission controller starts adding new agents one at a time. This continues till all the agents converge to the average of initial states (17) and thus proves to be effective.

3.6 Conclusion

In this paper, a novel optimal admission control and self-triggered consensus control co-design has been developed for networked MAS with unknown system dynamics. To better optimize the MAS physical system performance and network resource simultaneously, a novel networked MAS co-model has been developed

that can effectively present the networked MAS from both control and network aspects. Next, adopting the emerging reinforcement learning technique, a novel distributed optimal co-design has been designed that could not only optimize the self-triggering time, MAS consensus control, and also maximize the number of agents that can access the network. Due to the feature of reinforcement learning technique, proposed algorithm can relax the requirement about networked MAS system dynamics and further improve the practicality.

3.7 References

1. Hong Y (1998) Networked Control Systems. [Online document] available http://www.enme.umd.edu/icelab/ncs/ncs.html

2. Gupta R.A., Chow MY. (2008) Overview of Networked Control Systems. In: Wang FY., Liu D. (eds) Networked Control Systems. Springer, London.

3. Y. Halevi and A. Ray. "Integrated communication and control systems: Part I—Analysis", *J. Dynamic Syst., Measure. Contr.*, vol. 110, pp. 367-373, 1988.

4. H. Shousong and Z. Qixin, "Stochastic optimal control and analysis of stability of networked control systems with long delay", *Automatica*, vol. 39, 2003, pp. 1877-1884.

5. P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks", *Automatic Control, IEEE Transactions on*, vol. 52, no. 9, pp. 1680-1685, sept. 2007.

6. M. Mazo Jr., A. Anta, and P. Tabuada, "On self-triggered control for linear systems: Guarantees and complexity", *European Control Conference*, 2009.

7. E. Garcia P.J. Antsaklis. "Model-based event-triggered control with time-varying network delays", *50th IEEE Conference on Decision and Control, Orlando*, 2011.

8. D. Lehmann, J. Lunze. "Event-based control with communication delays and packet losses", *International Journal of Control*, 85(5):563- 577, 2012.

9. Lakshman, T.V., Mishra, P.P., and Ramakrishan, K.K., "Transporting compressed video over ATM networks with explicit-rate feedback control", *IEEE/ACM Transactions on Networking*, Vol. 7, No. 5, October 1999.

10. Jain, R., “Congestion control and traffic management in ATM networks: recent advances and a survey”, *Computer Networks and ISDN Systems*, Vol. 28.

11. Liew, S. and Yin, D.C., “A control-theoretic approach to adopting VBR compressed video for transport over a CBR

12. communications channel”, *IEEE/ACM Transactions on Networking*, Vol. 6, No. 1, 42-55, February 1998.

13. Jagannathan Sarangapani, “Wireless Ad Hoc and Sensor Networks”, *CRC Press*, pp 170-210, 2007.

14. Bae, J.J. and Suda, T., “Survey of traffic control schemes and protocols in ATM networks”, *Proceedings of the IEEE*, Vol. 79, No. 2, February 1991, pp. 170âĂŞ189.

15. Gurin, R., Ahmadi, H., and Naghshineh, M., “Equivalent capacity and its application to bandwidth allocation in high-speed networks”, *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 7, 968âĂŞ981, September 1991.

16. Jamin, S., Danzig, P.B., Shenker, S.J., Zhang, L., “A measurement-based admission control algorithm for integrated service packet networks”, *IEEE/ACM Transactions on Networking*, Vol. 5, No. 1, February 1997.

17. Chen, B., Zhang, Y., Yen, J., and Zhao, W., “Fuzzy adaptive connection admission control for real-time applications in ATM-based heterogeneous networks”, *Journal of Intelligent and Fuzzy Systems*, Vol. 7, No. 2, 1999.

18. Cheng, R.-G., Chang, C.-J., and Lin, L.-F., “A QoS-provisioning neural fuzzy connection admission controller for multimedia high-speed networks”, *IEEE/ACM Transaction on Networking*, Vol. 7, No. 1, 111âĂŞ121, 1999.

19. Peng, M., Jagannathan, S., and Subramanya, S., "End to end congestion control of multimedia high speed Internet", *Journal of High Speed Networks*, 2006.

20. P. J. Werbos, "A menu of designs for reinforcement learning over time", *J. Neural Networks Contr.*, vol. 3, pp. 835-846, 1983.

21. Dimos V. Dimarogonas, E. Frazzoli and K Johansson, "Distributed Self-triggered Control for Multi-agent Systems", *49th IEEE conference on Decision Control*, pp. 6716.

22. F. L. Lewis and V.L. Syrmos. Optimal Control. 2nd ed., Wiley, New York, 1995.

23. H. Xu, Luis R. G. Carrillo, "Near Optimal Control and Network Co-design for Uncertain Networked Control System with Constraints", *American Control Conference*, pp. 2339-2344, May 2017.

24. H. Xu, and S. Jagannathan, "Model-free Q-learning over Finite Horizon for Uncertain Linear Continuous-time Systems", *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2014 IEEE Symposium Orlando. FL*, 2015

25. J. Y. Lee, J. B. Park, and Y. H. Choi, "Integral Q-learning and explorized policy iteration for adaptive optimal control of continuous time linear systems", *Automatica*, vol. 48, pp. 2850-2859, 2012.

26. H. Xu, S. Jagannathan and F. L. Lewis "Stochastic optimal control of unknown linear networked control system in the presence of random delays and packet losses", *Automatica*, vol. 48, pp. 1017-1030, 2012.

Chapter 4

System Overview

The methodologies proposed in this thesis in the first two chapters were tested outdoors using three octacopters and a ground vehicle. The consensus was achieved in position, heading and altitude. The system consists of many different components interacting with each other to make flying possible. Following 6 are the main components which are described here.

- **Airframe:** The octacopter drone used as agents.
- **AutoPilot:** Used for low level control of the airframe.
- **Companion Computer:** Onboard computer for high level decision making and control.
- **Communication Devices:** Radio devices used to exchange information between agents and base station.
- **Real time Kinematic Global positioning system:** High precision GPS receivers used for accurate positioning.
- **Ground Control Station:** Base station used for drone tracking.

4.1 Airframe

The airframe used in this project is the DJI S1000 frame. This frame is build from synthetic material call carbon fiber which is a light material with high tensile strength. The company DJI Inc. made this frame with robustness and ease of photography kept in mind. The figure 4.1 shows top-side view of the completed DJI frame octa-copter.

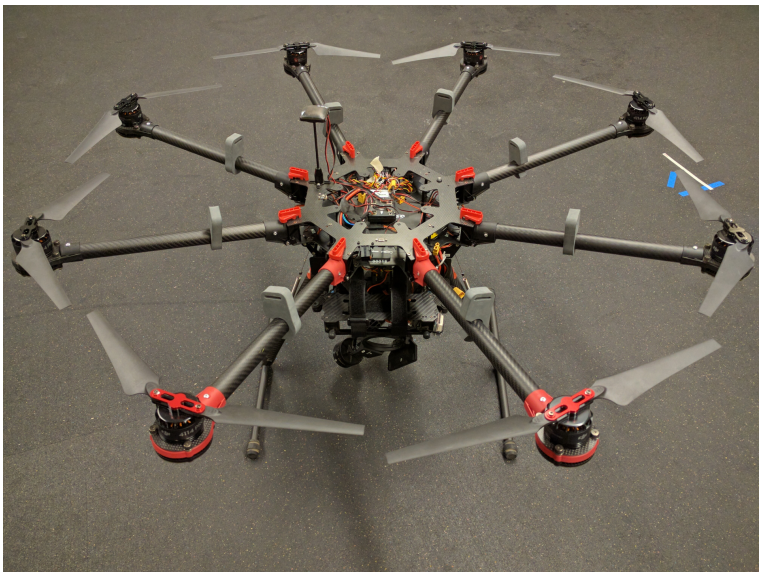

Figure 4.1: DJI S1000 Frame

4.2 AutoPilot Hardware: Pixhawk

Pixhawk is an open-hardware project. It is widely used by hobbyists and industrialists alike. It features a main controller processor and a math co-processor which delivers a high precision code completions. Figure 4.2 shows the pixhawk autopilot used with the DJI S1000 airframe.

Figure 4.2: Autopilot - Pixhawk

4.3 Companion Computer: Odroid XU4

ODROID-XU4 is a small form factor computing device capable of running a full fledged operating system. Further information about this device can be found at $http://www.hardkernel.com/main/products/prdt_info.php?g_code = G143452239825$

4.4 Communication Devices: DigiMesh XBee Modules

Xbee and Xbee-pro digiMesh 2.4 is copyright of digi family of communication devices. They are radio frequency modules which provides a peer-to-peer and mesh networking capabilities. These devices are micro devices used in embedded widely. They provide robust communications which is defined by their one of a kind routing capabilities for embedded networking.

Figure 4.4 shows the Xbee-pro digimesh module used in the project. The Antenna

Figure 4.3: odroid

used by this module produces more power and has a better range than the previous antenna xbee had.

4.5 Global Positioning Device: Piksi RTK GPS Module

Piksi RTK GPS modules are designed specifically for unmanned vehicles be it air, water or ground based. They provide a small form factor extreme precision GPS/GNSS functionality. The company also provides these in different sizes and forms. This project uses the 53x53mm form factor. The Piksi transceiver supports a high-performance, onboard Digital Signal Processor (DSP) and a flexible correlation accelerator. Figure 4.5 shows the swift navigation RTK GPS used for the project.

Figure 4.4: Xbee-pro Digimesh 2.4 GHz

4.6 Ground Control Station

The ground control station used for the project is the Mission Planner. This software is freely available under LGPL v2 liscense. This software is used to visualise the position of the aircraft and it can also provide some high level commands as takeoff, land , GPS locations and basically all path planning details. To convert the Mavlink frames to machine understandable frames, a companion microcontroller is used in conjunction with the Ground control station. This microcontroller is Arduino Mega 2560. Figure 4.6 shows Arduino Mega 2560 used in the project.

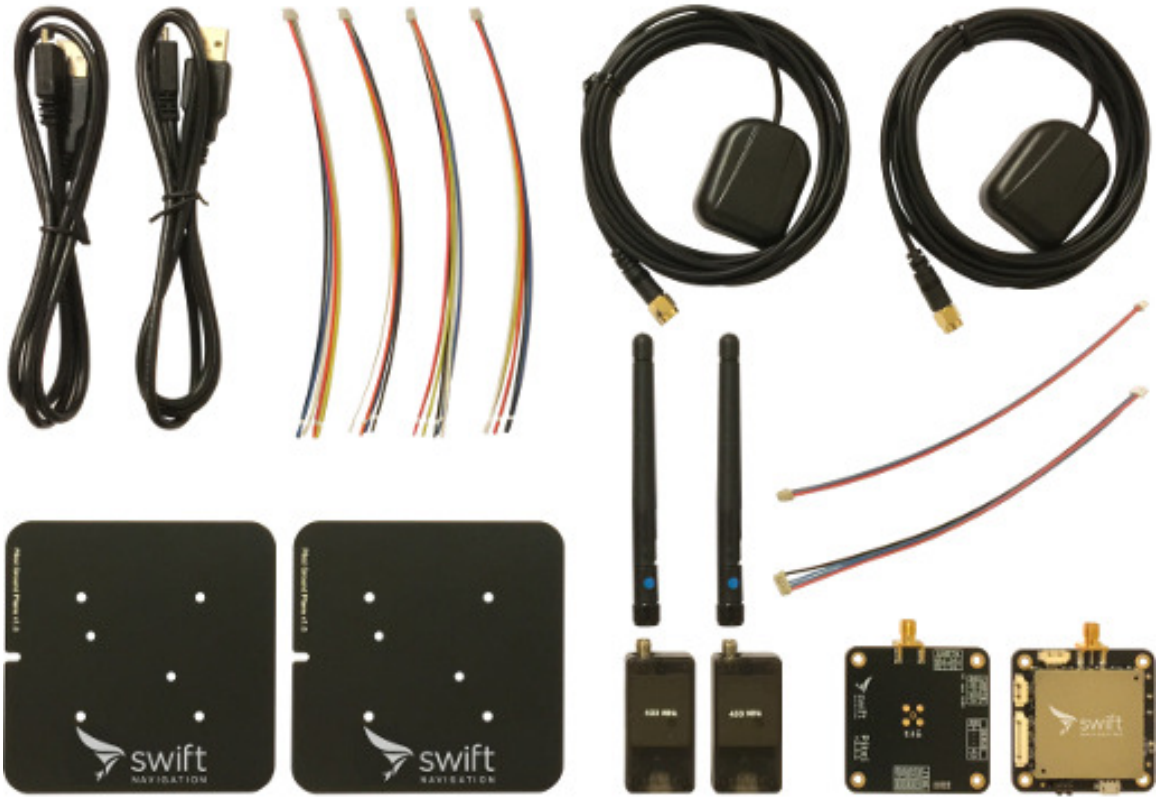

Figure 4.5: RTK GPS

4.7 System Integration

For the purposes of this project, 3 DJI S1000 frames were used. Each system is exactly the same except the sensing capability. One of the drone is equipped with HD camera and gymbal, while others are equipped with a thermal sensor and a wireless heartbeat sensor. Figure 4.7 shows the system integration. The figure shows an extra Octacopter with Lidar, this will be researched upon in future.

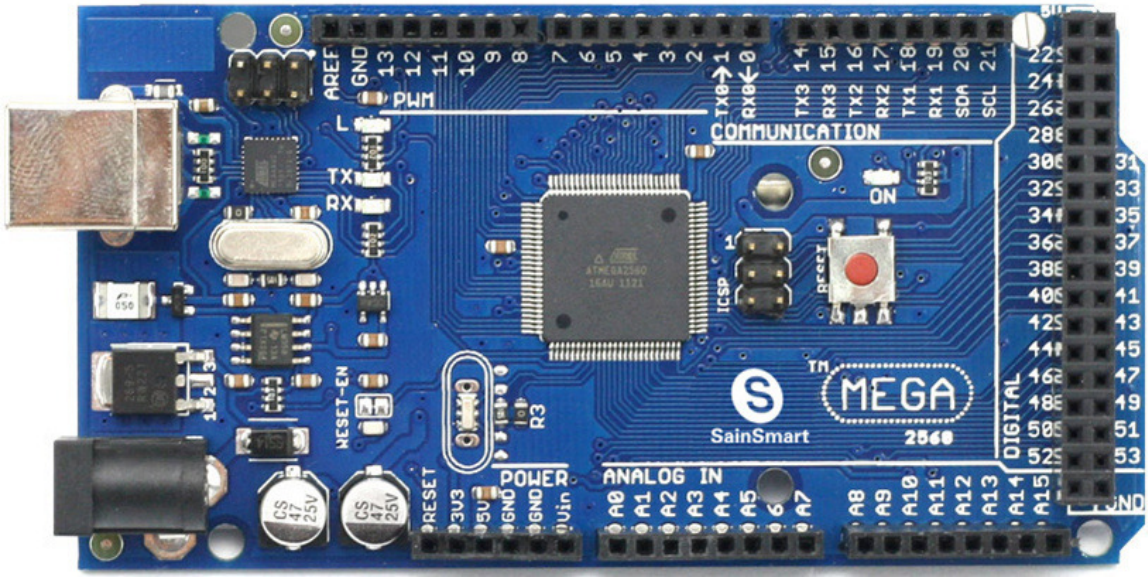

Figure 4.6: Arduino Mega

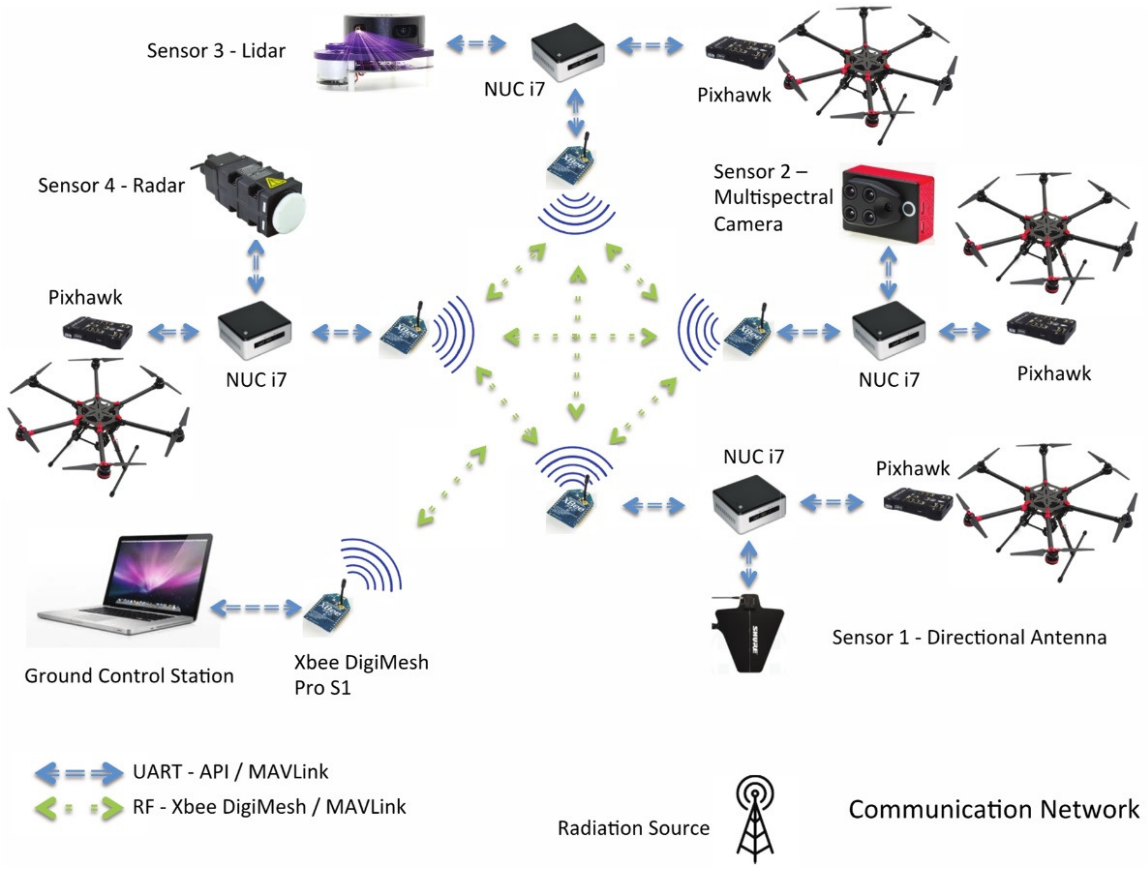

Figure 4.7: Multiple Agents Octacopter

4.8 Complete System Integration

The software consists of two parts i.e. **Communication Matrix** and **MAVLink Xbee encapsulator**. These have been developed in Ubuntu 14.04 LTS. Remainder of the chapter elucidates the contents and design of the software. In most cases the design is only a means to allow the reader to understand the contents more easily.

4.9 Defining the Communication Model

4.9.1 Prerequisites

1. **Python 2.7.6** Make sure the system has python binaries installed. To check the installation and version of the installation open a terminal and run the following command:
   Command: *python - -version*
   If user get output which is written as *python 2.7.6* or greater one is good to go.
   If not then run the following command:
   Command: *sudo apt-get install python*

2. **python-Serial** Make sure the system has proper python-serial binaries installed. This program relies on python-Serial to send and receive messages over the Radio and XBEE communication channel. To check the installation run the following command: **Command:** *sudo apt-get install python-serial*
   It will tell if one needs to install or it is already installed.
   There is a package named python3-serial available in the ubuntu repository. Do not install this if the system has python 2.7.6 installed. If user has python3 installed then only install python3-serial on to the system.

4.9.2 Graphical User Interface for Communication Matrix

This part of the software is the first step that will get the communication working in a user defined topology. It has to be run where Ground Control Station is installed. To run this part of the software make sure one has completed the **??** Prerequisites.

To use this software one needs to define the communication model of the multiple agents. Hypothetically lets say user wants 3 agents to work with. Arrange them as if one is arranging a 3x3 matrix as follows:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

One can add multiple agents to the software right at the first screen as shown in figure 4.8. Lets assume user wants to add 3 agents as discussed earlier.

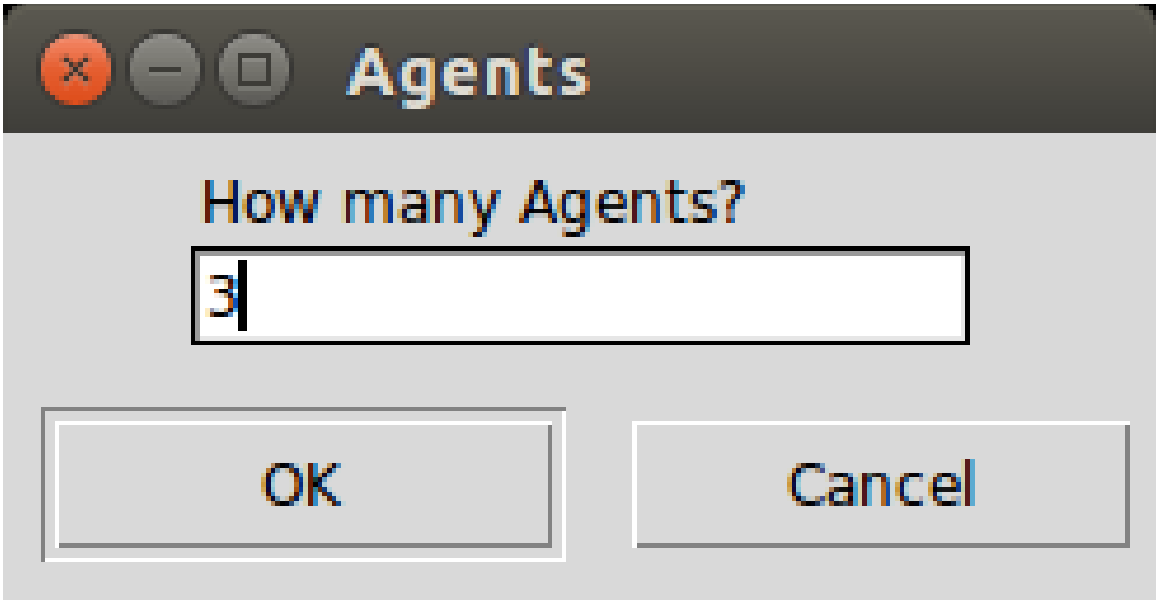

Figure 4.8: Communication Matrix GUI add agents

Further if one wants to have a topology where communication between agents 2 and 1 is bidirectional and communication between 3 and 1 is unidirectional with 3

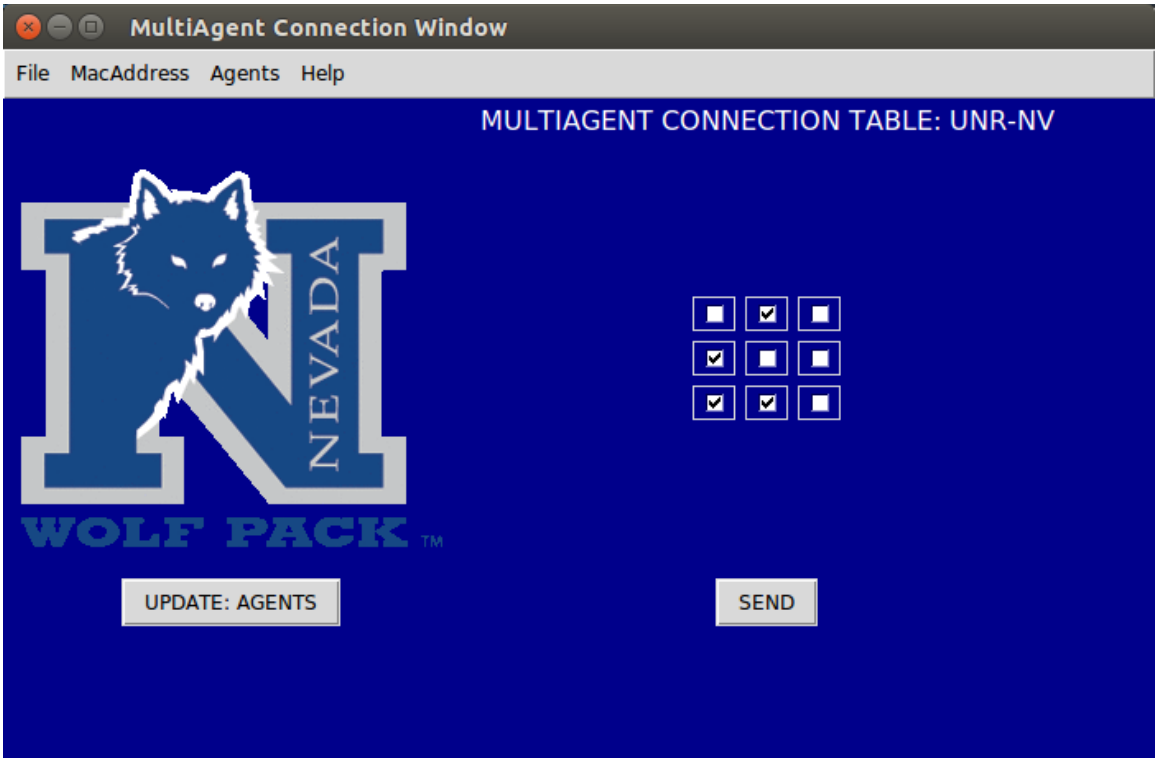

Figure 4.9: Communication Matrix GUI

sending to 1 and communication between 3 and 2 is unidirectional with 3 sending to 2 then intuitively one would want to enable A21, A12, A31 and A32 from the matrix. Needless to say A11, A22 and A33 will not make any difference if enabled or disabled.

The same is true with Communication matrix for this software. Figure 4.9 will show the exact scenario as discussed.

Different agents are assigned different number with the help of ini file. In the software all the agent mac addresses are added serially. One can add all the agent's mac addresses as follows:

As shown in figure 4.10. One can add different mac addresses via **ADD** button in the file menu.

After selecting the **ADD** button in file menu, one will see default text editor opened and user can add all the mac addresses as shown in figure 4.11.

Make sure one has the mac address written under agent number with every 8 bits separated by a colon (:). The number of mac agents specified in step 1 overrides the number of agents specified in mac addresses file (ini file). And the first 8 bits define the agent number followed by space. Please refer to figure 4.10. Messing up the file format in any way will render the communication system unpredictable.

If user does not specify the communication model the default communication is fully enabled. That is all the agents will communicated with one another bidirectionally.

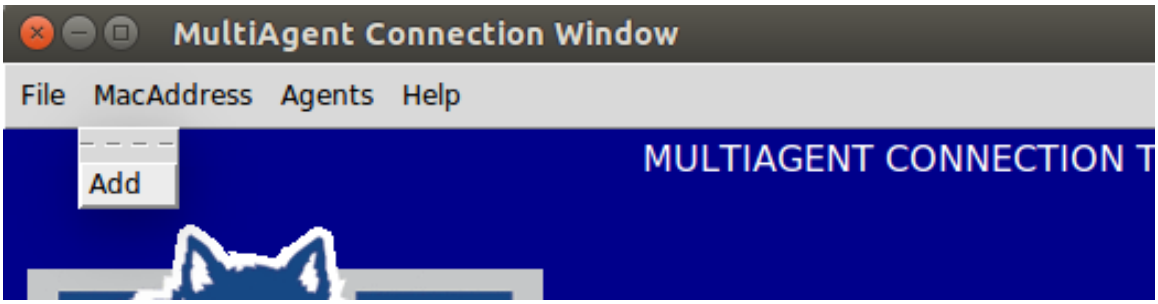

Figure 4.10: Communication Matrix GUI add mac

Once user has decided the topology and set the mac addresses correctly, press the button **Send**.

The software will configure the agents and ground control station to communicate in the defined topology. If in case user wants to change the number of agents in between the experiment press **Update: Agents**. This will take user to the first screen as depicted in figure 4.8. Then follow the steps as described above.

4.10 MAVLink Xbee Encapsulator

This software is part of companion computer of the UAV. It is responsible for communicating between pixhawks and ground control station. The ground control station that is available on internet i.e. *Mission Planner* for windows or *APM Planner*

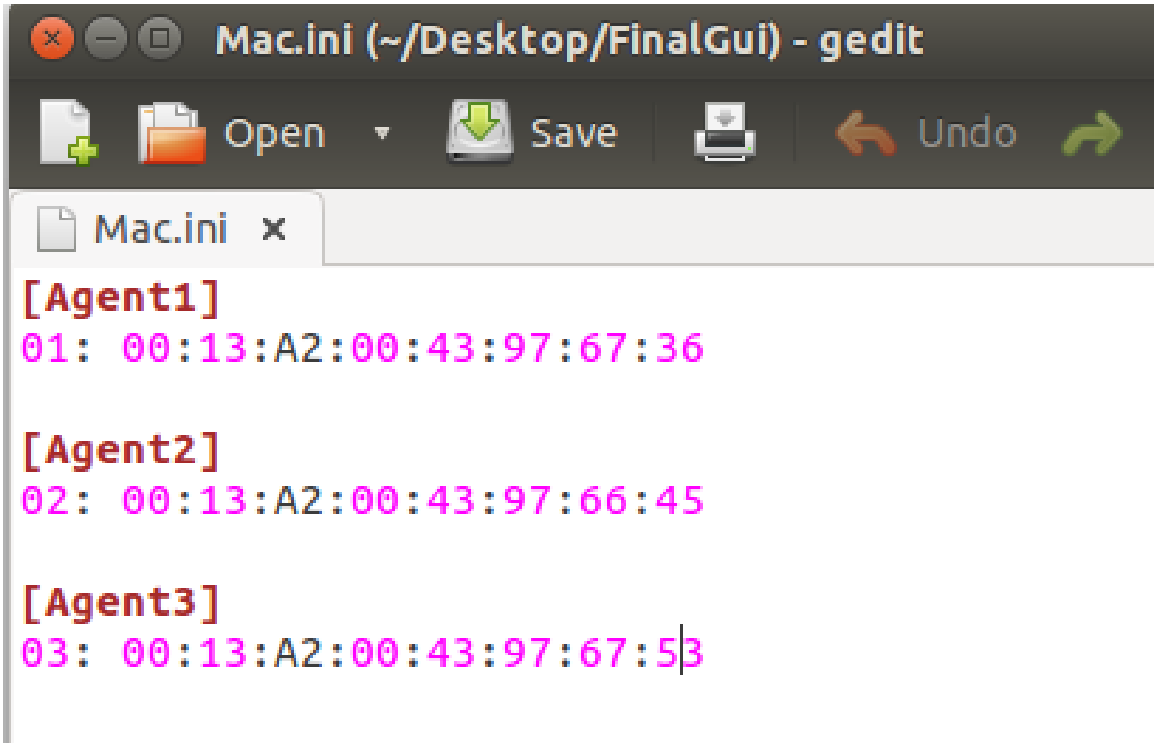

Figure 4.11: Add mac

for ubuntu are Single Vehicle capable (with a reduced support to show multiple vehicles) as of when this document was written. To enhance the multiple vehicle displaying capabilities this software uses XBEEs with Digimesh protocol. This protocol theoretically allows upto 65536 maximum agents to communicate simultaneously.

4.10.1 MAVLink Protocol

MAVLink is short for Micro Air Vehicle Link. It was created by Lorenz Meier in 2009 to cater needs of communication between small unmanned vehicles be it air, ground or water based. Pixhawks and similar commercial Auto-Pilots are using MAVLink protocol as a sole communication mode with different Ground Control Stations. It has already been widely deployed and is still being researched upon for new applications.

Frame Structure

As already mentioned MAVLink is a protocol which is a header-only message library; it works solely as an effective end to end message delivery system. The communication happens as encoded stream of bytes sent serially via radios. Each mavlink message varies between 10 bytes to 68 bytes. Following Table shows the structure of Mavlink packets.

Table 4.1: Mavlink Frame Structure

Frame Structure		
Field Name	*Index (Bytes)*	*Notes*
Start-of-frame	0	Denotes start of frame 0xFE
Payload-length	1	Length of Payload (n). Rolls back after 255
Packet Sequence	2	This is used to detect packet loss and retransmissons
System ID	3	Identification for Physical SYSTEM
Component ID	4	Identification for Internal subsystem e.g. IMU or GPS
Message ID	5	Identification for decoding messages correctly
Payload	6 to (n+6)	The data which is to be sent (encoded and decoded)
CRC	(n+7) to (n+8)	Checksum of the entire packet except the Start of frame delimiter. LSB First

Table 4.2: Mavlink Frame Structure

- **System ID:** This number is a unique number defining the source and destination. This is included in dispatch a message to the controller pixhawk via radio or USB port.
- **Message ID:** This field tells what is a particular message about.

- **Component ID:** This is planned for future presently not in use. This is a subsystem inside the main system.
- **CRC:** This field is used for integrity checking. It is calculated using part of the messages and added as last two bytes in to the sending message itself. ITU X.25/SAE AS-4 hash standard is used to calculate the CRC which includes Start-of-Frame indicator.
- **Payload:** This field contains the actual data that will be used to communicate between the Pixhawk and Ground Control station. A seed value is also added to the end of the messages when calculating the CRC. This seed is different for every message.

Protocol Processing:

The mavlink packet is a data pack that contains a non-constant number of bytes for different messages. Pixhawk gets streaming bytes via radio modules, gets into the physical stack via. radio and decodes the message in software stack. The packets contain the payload which is needs to be extracted. Following are the steps the code need to do to extract the information.

1. The code interpreter has a method called handlemessage. This is present in ArduPilot/arducopter folder in `GCS_Mavlink.cpp`.
   It basically decodes the System ID and Component ID from the packets. Any system using Mavlink has a System ID and a Component ID. The Component ID is for a subsystem attached to Pixhawk.

2. Extraction of payload data from the message is done and put into data packets. This data structure is based on a type of information.

3. A data structure is formed by putting together the data. There are numerious types of data structures e.g. for RC channels, GPS positioning etc. that puts together same things together.

Ground Station Control to Quadcopter:

After encoding or decoding **System ID** and **Component ID**, the software moves to encoding or decoding the **Message ID** `MAVLINK_MSG_ID_` type and once the message is detected, the **payload** information is decoded and used. Following are some of the Message IDs used for the purposes of this project

1. `MAVLINK_MSG_ID_GPS_RAW_INT #24` This message encapsulates global position. This is a raw sensor value and not global position estimate of the system. It has information related to latitude, longitude and altitude.

2. `MAVLINK_MSG_ID_GLOBAL_POSITION_INT #33` This message encodes the latitude, longitude, altitude, heading and GPS based ground speed.

3. `MAVLINK_MSG_ID_GPS_RTK #128` This message encodes Piksi GPS receiver's information. Same as the GPS INT message, it contains the latitude, longitude and altitude information.

4. `MAVLINK_MSG_ID_ATTITUDE #30` This message encapsulated the roll, pitch, yaw, rollspeed, pitchspeed and yawspeed information.

5. `MAVLINK_MSG_ID_ADSB_VEHICLE #246` This message give the location and informaiton of ADSB vehicle. It has following information encapsualted: ICAO address, latitude, longitude, altitude type, altitude, heading, horizontal velocity, vertical velocity, call sign, emitter type, time since last communication in seconds, flags and squawk code.

Copter to Ground Control Station

The Ground Control Station is like a mediator between the operator and the copter. Copter sends the information and GCS displays them on the screen.

Following is a code snippet which decodes the byte stream in this project software:

```
if (px4_intf.read().encode('hex') == 'fe') and (true_stx == True):
true_stx = False
#print "Got MAV_STX"
mav_len = (px4_intf.read())
b = mav_len.encode('hex')
#print "mav len : " + (mav_len)
a = int(b,16)
#print "lenght is : " + str(a)
mav_seq = (px4_intf.read())
mav_sysid = (px4_intf.read())
mav_compid = (px4_intf.read())
#print "component id is :" + str(int((mav_compid).encode("hex")))
mav_msgid = (px4_intf.read())
#print "message id is :" + str(int((mav_msgid).encode("hex"),16))

for i in range(0,a):
frame_mav = (px4_intf.read())
mav_payload += frame_mav
```

4.10.2 Arduino - Encapsulator and Decapsulator

The main objective of using an embedded microcontroller connected to GCS is to encapsulate the API headers to the existing Mavlink frames from and to the GCS. For the purposes of this project Arduino Mega 2560 was selected. The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

Arduino boards support serial library where the digital input/output pins can be converted to UARTs (hardware serial ports). During the initial testings it was found that making a digital input/output to a software serial port and switching it between the physical UART ports, an extra clock of processing is required which makes the mavlink frame skip a byte each time it switches between the physical UART port and the sotware serial port. To overcome this issue, multiple physical UART ports are required to be present on the embedded board. This led to selecting the Arduino Mega 2560 over the Arduino Uno board.

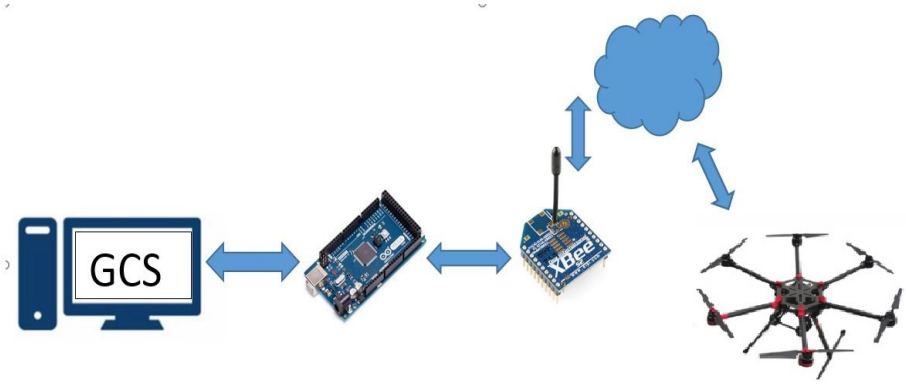

Figure 4.12: Arduino Mega Encapsulator Decapsulator

Figure 4.12 shows the process of encapsulating mavlink frames from a ground control station GCS to all the way up to the communication to the quadcopter.

As described in section 3.1 API headers can be added to arbitary data and sent over the XBee Digimesh network. Next in section 3.2 Mavlink frames were described where communication from GCS to Drones was described. The idea here is to intercept the communication between the GCS and Drone and add new communication channel as XBee API framings.

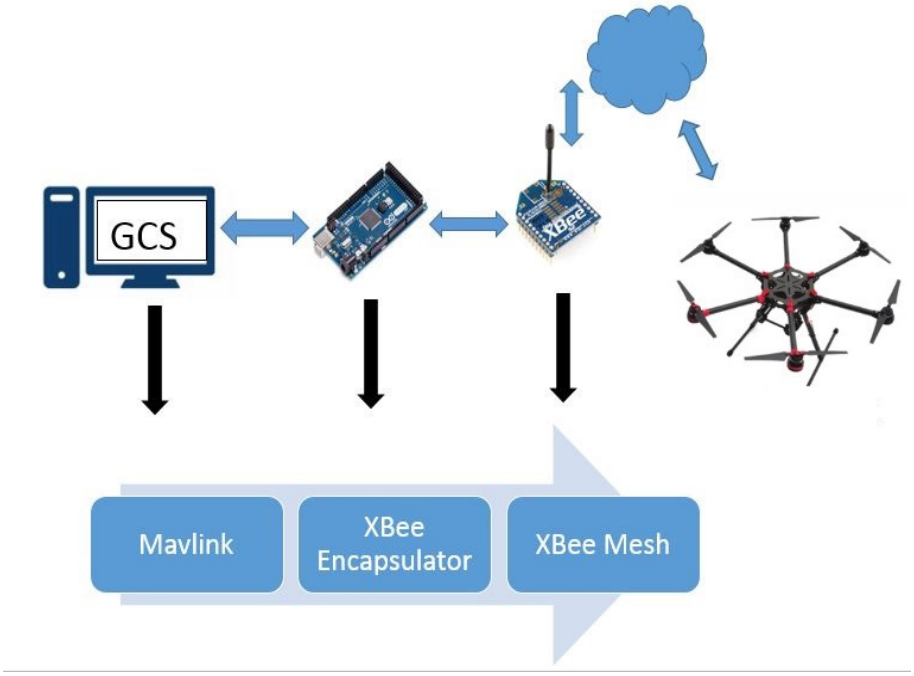

Figure 4.13: Arduino XBee Communication Process

Figure 4.13 shows the process of encapsulation. The decapsulation is just the reverse of this process. The Data from GCS is intercepted and each mavlink frame that starts with the start of frame delimiter till the two byte checksum (already discussed in section 3.2) of the frame is put inside the data frame structure of the API frame (discussed in section 3.1).

Following is a pseudocode which delineates the embedded processing of Arduino:

```
1. Initialize serial
```

```
2. Initialize api headers
3. Read from GCS
 a. Wait for Mavlink STX
 b. Read the data
 c. Encapsulate the mavlink data including stx with headers of step 2.
 d. Calculate CRC
 e. Add CRC as trailer
 f. Send via XBee serial port
4. Read from XBee interface
 a. Wait for api STX
 b. Read Data
 c. Decapsulate the API data structure from MAV STX till MAV CRC2
 d. Send via Serial Interface to GCS
```

4.11 Experimental Results

Experiments were performed at 2 places, one indoors at Department of Electrical and Biomedical Engineering, University of Nevada, Reno at 1664 N Virginia St, Reno, NV 89557 and the second one outdoors at Rancho San Rafael Park at 1595 N Sierra St, Reno, NV 89503.

The aim of the experiments was to visualise the locations of multiple Quad Copters on a Ground Control Station and log the data on the companion computer "Odroid".

Figure 4.14. illustrates the visualisation on a ground control station (GCS) called "APM Planner" which is a Linux based GCS freely available under LGPL v2 license.

As shown in Figure 4.14 two main components of the interface are QuadCopter

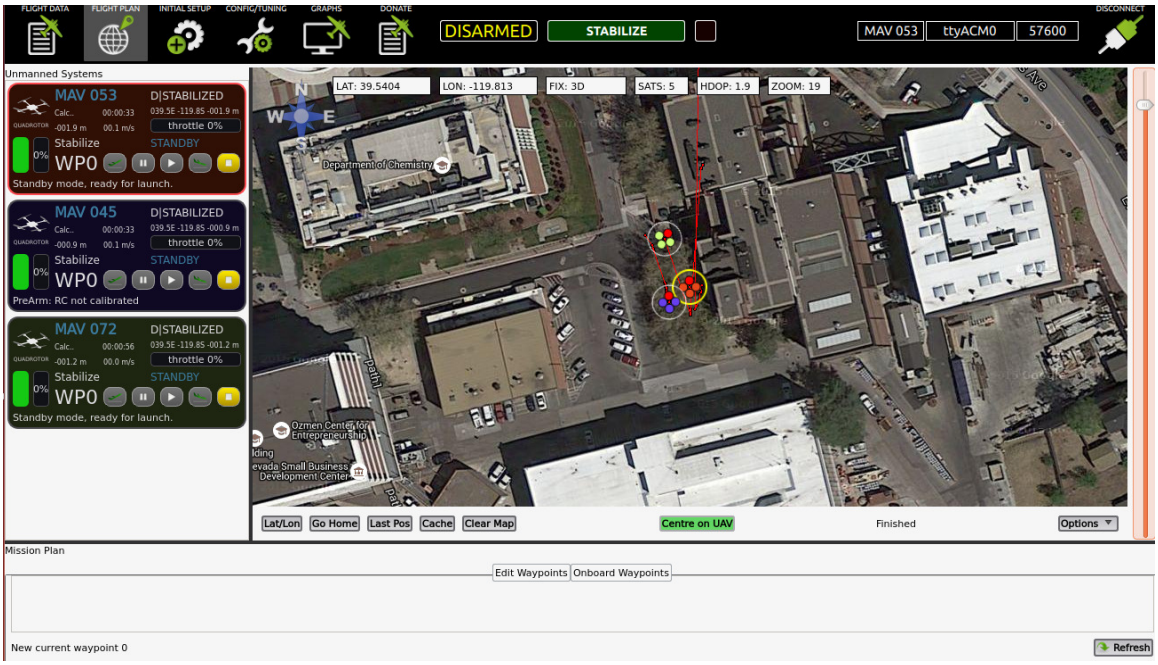

Figure 4.14: Indoor Test

List Pane and Map section.

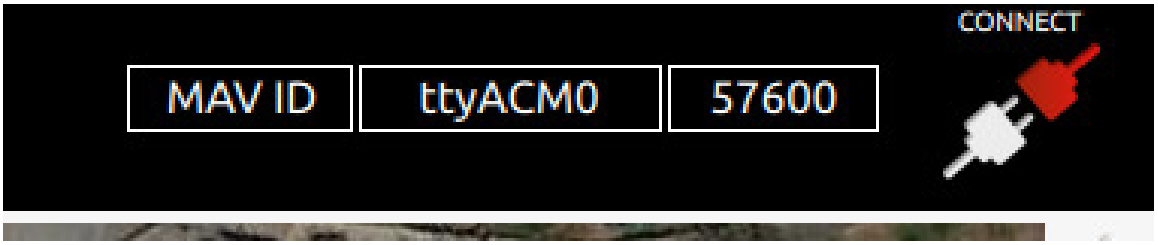

Figure 4.15: Disable Enable button for GCS

After starting the GCS user will need to connect the QuadCopter Radio interface by clicking the right top corner of the GUI, as show in Figure **??**. Default disconnected icon looks like the picture as shown on right side.

Once proper baudrate and ports are selected, the icon turns green and user will be able to see the copters on the map.

Figure 4.16 shows the Quadcopter List Pane. Users can click on any one of

Figure 4.16: Side Pane Multiple Quadrotors

the quadcopters and send control commands, if needed. It shows the name of the Quad-copters as shown "MAV 053", "MAV 045" and "MAV 072". Is also shows some basic information like speed, time since copter started, Copter Modes etc.
One important note on this pane is user can at any time shut off the Onboard Pixhawk from here.

Next important section of the GUI is the MAP and quad visualisation. Figure 4.17 below shows the MAP and the three copters.

The top side of the figure above shows the Latitude, Longitude and a few different information from GPS receiver. This information changes once user selects a different copter from the Quad Pane.

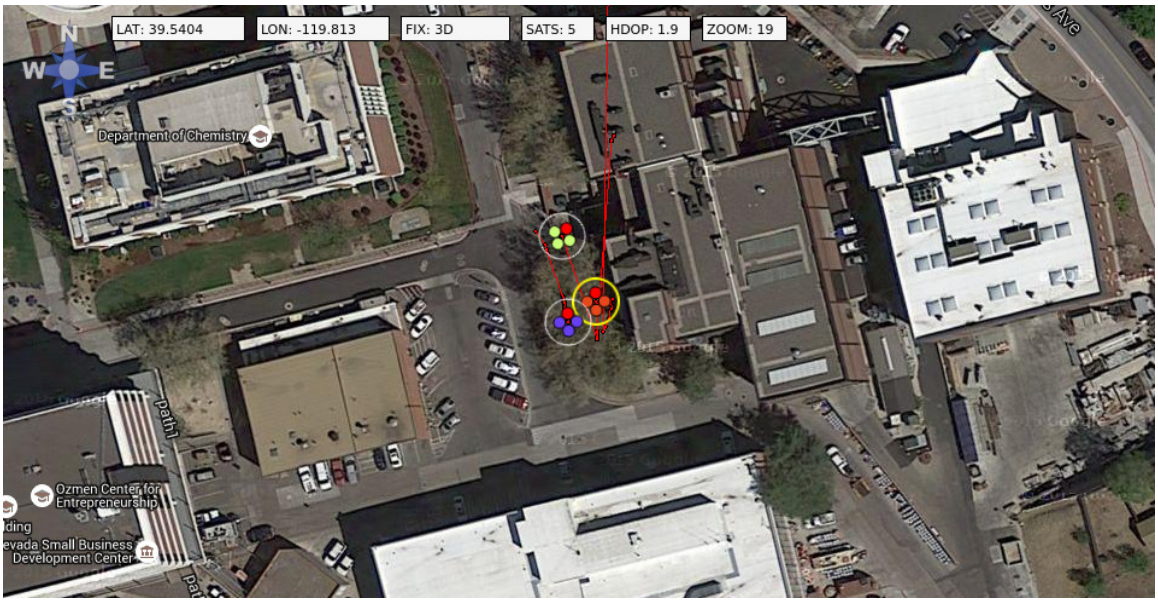

Figure 4.17: Ground control Station MAP Section

The other outdoor experiment that was performed in Rancho San Rafael was also based on getting visual conformation of GPS data being sent by copters and Logging the data on the companion computer "Odroid".

Screenshot 4.18 below shows the same experiment as above outdoors.

In this experiment the quad copters were kept stationary at different places.

Following are snap snapshots of the CSV log file from the Companion Computer Odroid:

Figure 4.18: Outdoors Experiment Test

Agent 1 Local				
Time	Latitude	Longitude	Altitude	Heading
18:53:33.806000	39.5468325	-119.832169	1376.576	145.24
18:53:33.806000	39.5468325	-119.832169	1376.576	145.24
18:53:33.806000	39.5468325	-119.832169	1376.576	145.24
18:53:33.806000	39.5468325	-119.832169	1376.576	145.24
18:53:33.806000	39.5468325	-119.832169	1376.576	145.24
18:53:33.806000	39.5468325	-119.832169	1376.576	145.24
18:53:33.806000	39.5468325	-119.832169	1376.576	145.24
18:53:33.806000	39.5468325	-119.832169	1376.576	145.24
18:53:33.806000	39.5468325	-119.832169	1376.576	145.24
18:53:34.305000	39.546832	-119.832155	1376.566	145.25
18:53:34.305000	39.546832	-119.832155	1376.566	145.25
18:53:34.305000	39.546832	-119.832155	1376.566	145.25
18:53:34.305000	39.546832	-119.832155	1376.566	145.25
18:53:34.305000	39.546832	-119.832155	1376.566	145.25
18:53:34.305000	39.546832	-119.832155	1376.566	145.25
18:53:34.305000	39.546832	-119.832155	1376.566	145.25
18:53:34.305000	39.546832	-119.832155	1376.566	145.25
18:53:34.305000	39.546832	-119.832155	1376.566	145.25
18:53:34.305000	39.5468322	-119.832157	1376.546	145.26
18:53:34.805000	39.5468322	-119.832157	1376.546	145.26
18:53:34.805000	39.5468322	-119.832157	1376.546	145.26
18:53:34.805000	39.5468322	-119.832157	1376.546	145.26
18:53:34.805000	39.5468322	-119.832157	1376.546	145.26
18:53:34.805000	39.5468322	-119.832157	1376.546	145.26
18:53:34.805000	39.5468322	-119.832157	1376.546	145.26

Figure 4.19: Data Log 1

Agent 1 Remote 1				
Time	Latitude	Longitude	Altitude	Headin
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22

Figure 4.20: Data Log 2

Agent 1 Remote 2			
Latitude	Longitude	Altitude	Heading
39.5468327	-119.832733	1390.036	333.76
39.5468327	-119.832733	1390.036	333.76
39.5468327	-119.832733	1390.036	333.76
39.5468327	-119.832733	1390.036	333.76
39.5468327	-119.832733	1390.036	333.76
39.5468327	-119.832733	1390.036	333.76
39.5468327	-119.832733	1390.036	333.76
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75
39.5468327	-119.832733	1390.046	333.75

Figure 4.21: Data Log 3

Agent 2 Local				
Time	Latitude	Longitude	Altitude	Heading
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:29.325000	39.5388928	-119.832	1439.212	44.21
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:30.525000	39.5403528	-119.82881	1437.988	44.24
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22
18:53:31.025000	39.5391428	-119.83221	1439.512	44.22

Figure 4.22: Data Log 4

Agent 2 Remote 1				
Time	Latitude	Longitude	Altitude	Heading
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24
18:53:26.025000	39.5459325	-119.832409	1376.534	145.24

Figure 4.23: Data Log 5

	Agent 2 Remote 2			
Time	Latitude	Longitude	Altitude	Heading
18:53:27.125000	39.5461327	-119.83311	1390.036	333.76
18:53:27.125000	39.5461327	-119.83311	1390.036	333.76
18:53:27.125000	39.5461327	-119.83311	1390.036	333.76
18:53:27.125000	39.5461327	-119.83311	1390.036	333.76
18:53:27.125000	39.5461327	-119.83311	1390.036	333.76
18:53:27.125000	39.5461327	-119.83311	1390.036	333.76
18:53:27.125000	39.5461327	-119.83311	1390.036	333.76
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75
18:53:27.125000	39.5461327	-119.83311	1390.046	333.75

Figure 4.24: Data Log 6

Agent 3 Local				
Time	latitude	Longitude	Altitude	Heading
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:31.125000	39.546807	-119.832228	1439.862	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16
18:54:32.225000	39.5468068	-119.832188	1439.902	44.16

Figure 4.25: Data Log 7

Agent 3 Remote 1				
Time	latitude	Longitude	Altitude	Heading
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:28.025000	39.5468434	-119.832131	1376.286	145.49
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.025000	39.5468487	-119.832132	1376.276	145.48
18:54:30.946000	39.5468497	-119.832133	1376.266	145.48

Figure 4.26: Data Log 8

Agent 3 Remote	2			
Time	latitude	longitude	Altitude	heading
18:54:29.025000	39.5468218	-119.832172	1391.166	334
18:54:29.025000	39.5468218	-119.832172	1391.166	334
18:54:29.025000	39.5468218	-119.832172	1391.166	334
18:54:29.025000	39.5468218	-119.832172	1391.166	334
18:54:29.025000	39.5468218	-119.832172	1391.166	334
18:54:29.025000	39.5468218	-119.832172	1391.166	334
18:54:29.025000	39.5468218	-119.832172	1391.166	334
18:54:29.025000	39.5468218	-119.832172	1391.166	334
18:54:29.025000	39.5468218	-119.832172	1391.166	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.025000	39.5468221	-119.832174	1391.146	334
18:54:30.525000	39.5468223	-119.832174	1391.146	333.99
18:54:30.525000	39.5468223	-119.832174	1391.146	333.99
18:54:30.525000	39.5468223	-119.832174	1391.146	333.99
18:54:30.525000	39.5468223	-119.832174	1391.146	333.99
18:54:30.525000	39.5468223	-119.832174	1391.146	333.99
18:54:30.525000	39.5468223	-119.832174	1391.146	333.99
18:54:30.525000	39.5468223	-119.832174	1391.146	333.99

Figure 4.27: Data Log 9

Agent 3 Local						
Time	latitude	Longitude	X	Y	Shifted X	Shifted Y
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:31.125000	39.546807	-119.832228	2449651.75	4062379.44258338	5.62361291	-1.5453687306
18:54:32.225000	39.5468068	-119.832188	2449648.78	4062379.42541261	2.6530398428	-1.5625394965
18:54:32.225000	39.5468068	-119.832188	2449648.78	4062379.42541261	2.6530398428	-1.5625394965
18:54:32.225000	39.5468068	-119.832188	2449648.78	4062379.42541261	2.6530398428	-1.5625394965
18:54:32.225000	39.5468068	-119.832188	2449648.78	4062379.42541261	2.6530398428	-1.5625394965
18:54:32.225000	39.5468068	-119.832188	2449648.78	4062379.42541261	2.6530398428	-1.5625394965
18:54:32.225000	39.5468068	-119.832188	2449648.78	4062379.42541261	2.6530398428	-1.5625394965
18:54:32.225000	39.5468068	-119.832188	2449648.78	4062379.42541261	2.6530398428	-1.5625394965
18:54:32.225000	39.5468068	-119.832188	2449648.78	4062379.42541261	2.6530398428	-1.5625394965
18:54:32.225000	39.5468068	-119.832188	2449648.78	4062379.42541261	2.6530398428	-1.5625394965

Figure 4.28: Data Log 10

Agent 3 Remote 1						
Time	latitude	Longitude	X	Y	Shifted X	Shifted Y
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:28.025000	39.5468434	-119.832131	2449643.23	4062382.57	-2.8983182516	1.5797098423
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.025000	39.5468487	-119.832132	2449643.12	4062383	-3.0111628235	2.0347348829
18:54:30.946000	39.5468497	-119.832133	2449643.16	4062383.11	-2.9720587116	2.1205886602

Figure 4.29: Data Log 11

Agent 3 Remote	2					
Time	latitude	longitude	X	Y	Shifted X	Shifted Y
18:54:29.025000	39.5468218	-119.832172	2449647.046	4062381	0.917041232	-0.27473219
18:54:29.025000	39.5468218	-119.832172	2449647.046	4062381	0.917041232	-0.27473219
18:54:29.025000	39.5468218	-119.832172	2449647.046	4062381	0.917041232	-0.27473219
18:54:29.025000	39.5468218	-119.832172	2449647.046	4062381	0.917041232	-0.27473219
18:54:29.025000	39.5468218	-119.832172	2449647.046	4062381	0.917041232	-0.27473219
18:54:29.025000	39.5468218	-119.832172	2449647.046	4062381	0.917041232	-0.27473219
18:54:29.025000	39.5468218	-119.832172	2449647.046	4062381	0.917041232	-0.27473219
18:54:29.025000	39.5468218	-119.832172	2449647.046	4062381	0.917041232	-0.27473219
18:54:29.025000	39.5468218	-119.832172	2449647.046	4062381	0.917041232	-0.27473219
18:54:30.025000	39.5468221	-119.832174	2449647.184	4062381	1.055322196	-0.24897605
18:54:30.025000	39.5468221	-119.832174	2449647.184	4062381	1.055322196	-0.24897605
18:54:30.025000	39.5468221	-119.832174	2449647.184	4062381	1.055322196	-0.24897605
18:54:30.025000	39.5468221	-119.832174	2449647.184	4062381	1.055322196	-0.24897605
18:54:30.025000	39.5468221	-119.832174	2449647.184	4062381	1.055322196	-0.24897605
18:54:30.025000	39.5468221	-119.832174	2449647.184	4062381	1.055322196	-0.24897605
18:54:30.025000	39.5468221	-119.832174	2449647.184	4062381	1.055322196	-0.24897605
18:54:30.025000	39.5468221	-119.832174	2449647.184	4062381	1.055322196	-0.24897605
18:54:30.025000	39.5468221	-119.832174	2449647.184	4062381	1.055322196	-0.24897605
18:54:30.525000	39.5468223	-119.832174	2449647.177	4062381	1.04825481	-0.23180528
18:54:30.525000	39.5468223	-119.832174	2449647.177	4062381	1.04825481	-0.23180528
18:54:30.525000	39.5468223	-119.832174	2449647.177	4062381	1.04825481	-0.23180528
18:54:30.525000	39.5468223	-119.832174	2449647.177	4062381	1.04825481	-0.23180528
18:54:30.525000	39.5468223	-119.832174	2449647.177	4062381	1.04825481	-0.23180528
18:54:30.525000	39.5468223	-119.832174	2449647.177	4062381	1.04825481	-0.23180528
18:54:30.525000	39.5468223	-119.832174	2449647.177	4062381	1.04825481	-0.23180528

Figure 4.30: Data Log 12

Chapter 5

Conclusion

In conclusion, this thesis presents theoritical approaches for event-triggered consensus control and self-triggered consensus control. In addition to these schemes, further development to them is also proposed. In case of event triggered consensus control, a novel distributed consensus control capable of handling a high order non-linearity with system uncertainities was presented. The proposed adaptive model was shown to have better performance than the traditional Zero Order Hold schemes. In case of self-triggered consensus control, a novel optimal network admission control co-design was presented. The proposed scheme utilized reinforcement learning technique, which takes care of the optimality condition for self-triggered consensus control. It also presented a novel network admission and consensus control scheme that can maximize the performance while keeping the stability criteria as a central theme.

In addition to theoritical simulations and results, this thesis presented a multi-agent drone control system both in hardware and software. The hardware presented was developed using state-of-the art drone system and accessories available at the time of writing this thesis and the software part was developed specifically for the purpose of testing the presented methodologies. However, as the presented

methodologies are of general nature, small tweaks are essential to mould this system for a number of different consensus control systems especially resource constrained embedded platforms such as drones. In the end, this thesis presented two successful missions done indoors and outdoors.

Chapter 6

References

1. Loianno, Mulgaonkar, Brunner, Ahuja, Ramanandan, Chari, Diaz, and Kumar, "A Swarm of Flying Smartphones", *in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 2016

2. Gupta R.A., Chow MY. (2008) Overview of Networked Control Systems. In: Wang FY., Liu D. (eds) Networked Control Systems. Springer, London.

3. Y. Halevi and A. Ray. "Integrated communication and control systems: Part I—Analysis", *J. Dynamic Syst., Measure. Contr.*, vol. 110, pp. 367-373, 1988.

4. H. Shousong and Z. Qixin, "Stochastic optimal control and analysis of stability of networked control systems with long delay", *Automatica*, vol. 39, 2003, pp. 1877-1884.

5. P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks", *Automatic Control, IEEE Transactions on*, vol. 52, no. 9, pp. 1680-1685, sept. 2007.

6. M. Mazo Jr., A. Anta, and P. Tabuada, "On self-triggered control for linear systems: Guarantees and complexity", *European Control Conference*, 2009.

7. E. Garcia P.J. Antsaklis. "Model-based event-triggered control with time-varying network delays", *50th IEEE Conference on Decision and Control, Orlando*, 2011.

8. D. Lehmann, J. Lunze. "Event-based control with communication delays and packet losses", *International Journal of Control*, 85(5):563- 577, 2012.

9. Lakshman, T.V., Mishra, P.P., and Ramakrishan, K.K., "Transporting compressed video over ATM networks with explicit-rate feedback control", *IEEE/ACM Transactions on Networking*, Vol. 7, No. 5, October 1999.

10. Jain, R., "Congestion control and traffic management in ATM networks: recent advances and a survey", *Computer Networks and ISDN Systems*, Vol. 28.

11. S. Jagannathan, Neural network of nonlinear discrete-time system, CRC Press, 2006.

12. E. Garcia, and P. J. Antsaklis, "Model-based event-triggered control with time-varying network delays", *in Proc. IEEE Conf. Decision and Contr.*, pp. 1650-1655, 2011.

13. M.S. Branicky, M.M. Curtiss, J. Levine and S. Morgan, "Sampling-based planning, control and verification of hybrid systems", *IEEE proceedings. Control theory and applications*, vol. 153-5, pp. 575-590, 2006.

14. R.W. Brennan ; M. Fletcher ; D.H. Norrie, "An agent-based approach to reconfiguration of real-time distributed control systems", *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, 2002.

15. J.H. Sandee, W.P.M.H. Heemels, P.P.J. v.d. Bosch, "Case studies in event-driven control", 2009.

16. A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-triggered control for discrete-time systems", *in Proc. Amer. Contr. Conf.*, pp. 4719-4724, 2010.

17. G.S. Seyboth, D.V. Dimarogonas, K.H. Johansson, "Event-based broadcasting for multi-agent average consensus", Automatica 49 (1) (2013) 245-252.

18. Garcia, E., Cao, Y., Wang, X., and Casbeer, D.W, "Cooperative control with general linear dynamics and limited communication: Periodic updates", *In Proc. IEEE ACC*, 3195-3200.

19. Huaipin Zhang, Dong Yue, Xiuxia Yin, Songlin Hu, Chun xia Dou, "Finite-time distributed event-triggered consensus control for multi-agent systems", *Elsevier Information Sciences*, vol 339, 20. pp. 132-142. 2016.

20. A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-triggered control for discrete-time systems", *in Proc. Amer. Contr. Conf.*, pp. 4719-4724, 2010.

21. P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks", *IEEE Trans. on Automat. Contr.*, vol. 52, pp. 1680-1685, 2007.

22. H. Su, G. Chen, X. Wang, and Z. Lin, "Adaptive second-order consensus of networked mobile agents with nonlinear dynamics", *Automatica*, vol. 47, no. 2, pp. 368-375, 2011.

23. Z.-G. Hou, L. Cheng, and M. Tan, "Decentralized robust adaptive control or the multiagent system consensus problem using neural networks", *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 3, pp. 636-647, Jun. 2009.